TRAFFIC LIGHT LEARNING AND PREDICTION

A Thesis

Presented to

The Academic Faculty

By

Madhuvanthi Sridhar

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Mechanical Engineering

Georgia Institute of Technology

May 2020

TRAFFIC LIGHT LEARNING AND PREDICTION

Approved by:

Dr. Berdinus Bras, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Richard Simmons
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Roger Jiao
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: [January 31, 2020]

ACKNOWLEDGEMENTS

First and foremost, I would like to recognize my mother and father without whom my accomplishments thus far would not have been possible. The support, love, advice, motivation, and unlimited encouragement that I have received from my parents, in addition to the multiple sacrifices they have made over the years, has allowed me to push my boundaries by pursuing higher studies and passionately research more into my field of Mechanical Engineering. I would like to especially thank Dr. Bert Bras (graduate research advisor) for his support and valuable advice throughout my research assignment. I would also like to thank Dr. Roger Jiao and Dr. Richard Simmons for serving as my thesis committee members. Additionally, I would like to acknowledge Dr. Wayne Whiteman, Dr. Aldo Ferri, Dr. Thomas Kurfess, Dr. Yogendra Joshi, Dr. Peter Hesketh, and Dr. Marc Smith, for their extremely useful career and research advice and informative class lectures throughout my undergraduate and graduate career. Last but certainly not the least, I would like to thank both of my uncles, Dr. A. Ramesh, Professor IIT Madras, Internal Combustion Engines and Institute Chair Professor & Chairman (CCE), and Mr. A. Vijayan, Director – Engineering, Research, and Development, Brakes India Private Limited, for the guidance and encouragement they have given me throughout my academic career in engineering at Georgia Tech.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

V2I	**Vehicle to Infrastructure**
V2V	**Vehicle to Vehicle**
GLOSA	**Green Light Optimized Speed Advisory**
SPAT	**Signal Phase and Timing**
DSRC	**Dedicated Short Range Communications**
ANN	**Artificial Neural Network**
Faster R-CNN	**Faster Regions with Convolutional Neural Network**
SSD	**Single Shot Detection**
YOLO	**You Only Look Once**
COCO (Microsoft Image Dataset)	**Common Objects in Context**
COM Port	**Communication Port**
IDE	**Integrated Development Environment**
UHD	**Ultra High Definition**
API	**Application Program Interface**
ADAS	**Advanced Driver Assistance Systems**
RAM	**Random Access Memory**

NOMENCLATURE

CLDB	**Central Learning Database Table - Stores Signal Records Uploaded During Learning Process**
CLDB Copy	**Central Learning Database Copy Table – Stores Signal Records (Historical) Uploaded During Learning Process**
TLIDB Table	**Traffic Light Information Database Table - Stores Information About Traffic Lights on a Driver's Route**
KB	**Knowledgebase – Stores Signal Length Predictions**
NBUSP	**Nightly Batch Updating Stored Procedure – Develops Signal Length Predictions Using Learning Data**

SUMMARY

Being stuck in traffic and not knowing when the signal will change when in a hurry to get to work is one of the worst situations to be in on the road. Although pedestrian countdown timers are useful in predicting when the signal will change, they are not available at all intersections and are slightly off in timing from when the actual signal changes. Further, signal lengths change throughout the day to accommodate varying levels of traffic, which is hard for even the smartest traffic light countdown timers and apps to predict without real-time signal transition data.

The goal of this research assignment is to implement a traffic light learning and prediction model in the transportation sector that allows cars to intelligently "learn" and "predict" traffic light behavior. There are many motivations to pursue research in this area, some due to technological advancements in the autonomous vehicle industry that encourage research in V2I/V2V communications to receive information about traffic lights real-time while driving (with or without a human driver). Other motivations include keeping a driver informed on the road to prevent accidents due to "missing a traffic light", reducing a driver's transportation costs and commute time by increasing fuel economy, energy efficiency, and vehicle part lifetime through idling prevention, and contributing to less damage to the environment by cutting down vehicle emissions.

The model developed in this research assignment can be divided into two main algorithms; one accomplishes "learning" traffic lights using a computer vision system, and the other uses the learned data to develop predictions on traffic signal lengths as well as provide other important information on the light back to the driver. The vision system developed in the first algorithm is a traffic light object recognition system that allows vehicles to detect three

different red and green light configurations as well as the green right arrow on live stream video capture. These state detections, along with the timestamp of detection, and the signal ID of the traffic light are uploaded as learning data into a central learning database table (CLDB) to be used later to form signal length predictions. The second algorithm can perform three separate functions that are integrated together to create a traffic light prediction system: 1. notifying the driver of the upcoming traffic light on their route, 2. determining the last known state of that particular traffic light (if it exists), and 3. offering a prediction on when the state of the traffic light will change if prior learning data exists at the time period of crossing the light. All of these functions are performed simultaneously, displaying updates to the driver on a GUI when in proximity to the traffic light. The predictions are formed by applying a nightly batch updating store procedure (NBUSP) to the newly uploaded learning data. All of the records are stored in a knowledgebase (KB) that is contacted real-time to provide the prediction to the driver when passing through the light at a known signal length prediction time period. This research methodology builds upon the leader/follower approach introduced in the prior work for this assignment, defining cars having access to a camera contributing to the "learning" process as "leaders", and cars only capable of receiving live updates as "followers." Leaders are also considered to be followers, but a follower cannot be considered as a leader.

The learning system has proven to be extremely accurate in identifying all signal configurations on live stream video capture without mistaking other similar objects on the road to be traffic lights. The prediction system generates and relays highly accurate signal length predictions to the driver using acquired learning data. Future work will introduce the ability to offer logical signal length predictions even if no prior learning data exists for a traffic light.

CHAPTER 1. INTRODUCTION

This chapter provides an introduction to the topic presented in the paper as well as motivations to pursue research in this area, a briefing of the approach taken in this research document to provide effective solutions to this issue, and a summary of the accomplishments and shortcomings of the implemented research methodology. The chapter closes with unveiling the order in which the proposal will be presented.

1.1: Technology of the Future - Autonomous Vehicles and V2I/V2V Communication

With a rapid advent in technology, more and more processes are being automated to relieve human workers of performing daily monotonous, and perhaps dangerous tasks. The automation industry is reaching its peak particularly in vehicular applications, with the development of self-driving cars by companies like Google, Audi, etc., that utilize little to no human input and are able to drive in all types of conditions with judgement levels equal to and at times surpassing humans. Along with self-driving cars, the ability for cars to "talk to one another" and with infrastructure (primarily communicator devices attached to traffic lights) is also gaining more interest, and researchers across the world are spending more time and efforts to develop inexpensive yet effective V2V and V2I technologies that can benefit drivers on the road. Figure 1 shows a diagram representing these technologies.

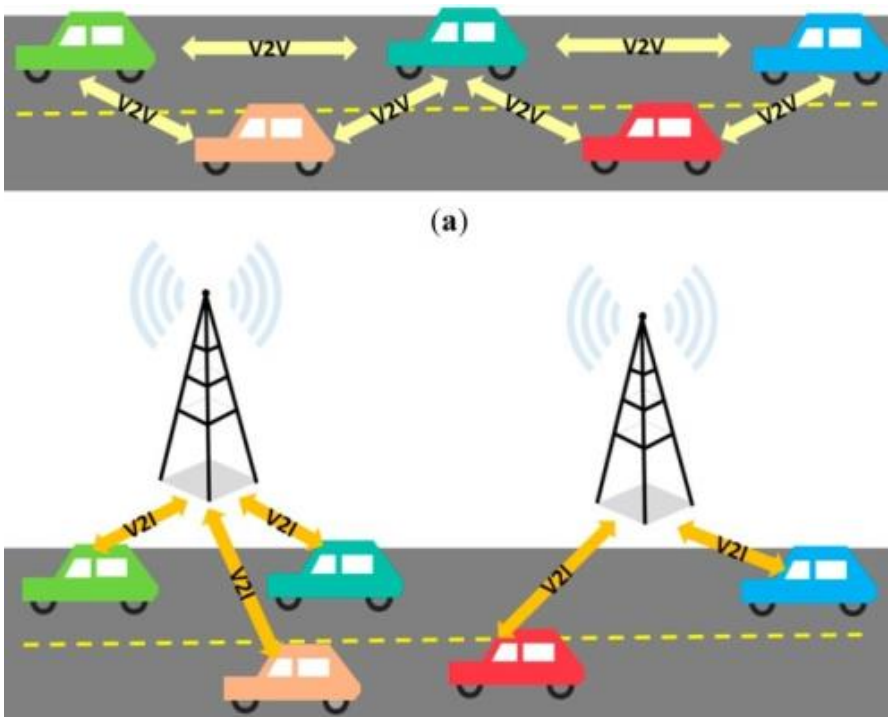

Figure 1 - V2I/V2V Communications

1.2: Keeping Drivers Informed – Accidents Due to Distraction and Careless Driving

Why is current research in the transportation engineering sector focused on making cars communicate in some form with traffic lights? Traffic lights serve to control traffic flow and manage intersections where cars are approaching from different directions. Multiple traffic regulations have been enforced by the law throughout the US and across the world for drivers passing through intersections, in order to prevent accidents due to multi-vehicular collisions, but many drivers still fail to heed these laws and do not take the punishments that follow from their disobedience seriously. Due to this recurring phenomenon, over 2.5 million intersection accidents are reported by the Federal Highway Administration annually (as updated in 2018), mentioned in [3]. But drivers do not just miss traffic lights due to negligence. External distractions, such as loud music, conversations with family members on the phone or in the car, and work tension or stress causing attention to be diverted from the road, are some of the leading

causes of road accidents at intersections. Developing a system that can provide information on current traffic light status as well as signal phase timings can help drivers be more informed and stay alert about traffic lights on their route.

1.3: Raising Vehicle Fuel Efficiency, Energy Consumption, and Part Lifetime While Reducing Traffic Congestion and a Driver's Commute Time

An intelligent traffic light learning and prediction system also has advantages in improving the fuel efficiency and energy consumption of a vehicle. Engine idling at traffic signals while waiting for a red light to turn to green, is a major contributor to decreasing the fuel efficiency of a vehicle and also wasting energy. Decelerating to a stop, idling the engine while at rest, and accelerating back to full speed additionally cause the vehicle to consume large amounts of energy while damaging important parts such as spark plugs, timing belts, etc. Fuel and energy wastage are especially important metrics to the average driver; spending money multiple times a week to fill up the vehicle's gas tank in order to travel long distances to work increases a driver's overall transportation costs immensely, making it harder for them to use their personal vehicles to commute every day. Further, replacing parts crucial to a vehicle's functionality is an extremely time and money consuming process that cannot be done multiple times a year.

Traffic congestion is another problem that is often experienced by drivers at prime intersections for a city or suburb. During peak hours of travel, drivers are required to wait for a long time in order for a signal to clear up. Developing a system that can provide updates to the driver on when the light will change state is useful to help them determine an optimal speed profile (within state mandated speed limits) to pass as many green lights as possible on their

route. This will allow signals to be less congested and promote free traffic flow, reducing individual commute time.

1.4: Cutting Down on Vehicle Emissions and Reducing Environmental Pollution

Environmental pollution from the transportation sector has been increasing vastly over the years. [43] claims that experiments have shown the concentration of polluted particles in the air to be 29 times higher at traffic lights than at other free traffic flow conditions. This is primarily due to the engine idling process that causes fuel to be wasted and toxic emissions to be released to the environment, damaging air quality and causing widespread pollution. Providing drivers information on the time remaining for the current state to change will allow them to make an educated decision on stopping the engine when approaching a red light, preventing damage to the environment due to excessive idling.

1.5: Briefing of the Research Approach

To address the research topic at hand, two interdependent algorithms were formulated, one to learn traffic lights and the other to use the learned data to offer predictions to the driver on approximate traffic signal phase lengths. In order to learn the traffic lights, a computer vision system running a traffic light object recognition model was developed. The trained model has the ability to detect and learn traffic lights in urban as well as suburban areas, since it was primarily trained on Atlanta (highly populated city) and Cumming (lower population suburb) traffic light images. The model is run on live stream video captured throughout the drive to perform continuous traffic light detection. A video frame processing traffic light state detections found by the model is represented in Figure 2.

Figure 2 - Red Light State Detections Performed on Live Stream Video

To predict and relay traffic signal lengths as well as other important information to the driver, multiple independent modules were developed and integrated together. First, the nearest traffic light to a driver's location is found by continuously calculating the distance between the driver's current location coordinates and the geographical coordinates of traffic signals on their travel route. Once the car approaches the light with close proximity, the signal ID as well as last known state of the signal is shown to the driver on a GUI update. Simultaneously, the signal ID, state (s) of the traffic light, and timestamp (s) at which the state (s) were encountered are continuously uploaded as detected by the model into the central learning tables in the form of signal records. The NBUSP is applied to the newly uploaded signal records to create signal length predictions. This batch procedure implements a self-refining algorithm that organizes

signal length prediction records by traffic light state and time of upload, with the underlying assumption that traffic lights do not change twice within a particular time interval of the same observed state (10 seconds for the assumptions of this test). These predictions are stored for each respective signal ID in the KB. Upon travelling through a light, if a prediction is found to be available at the current time, it is relayed back to the driver on the same GUI update offering the upcoming signal ID and last known state information for the light.

1.6: Organization of Thesis

The remainder of the document will be presented in the following order: Chapter 2 presents an overview of existing literature on the research topic. Chapter 3 details the research methodology, breaking the overall algorithm into multiple sub modules to explain their individual functions, and then combining it back together to present the big picture. Chapter 4 outlines and discusses the results obtained from each live sub module execution as well as a live on the road test of the overall algorithm. Chapter 5 makes reference to future work that can be done in this field of research. Chapter 6 concludes the document with final comments on the system and questions that are left open at the end of this assignment.

CHAPTER 2. BACKGROUND

A lot of recent research has focused on developing traffic light object recognition systems, vehicle countdown timers, etc. to predict traffic signal durations and reduce fuel economy due to idling at signals. This chapter will present a few of the most recent approaches taken to address this research problem, focusing individual attention on the efforts discussed above. It will start with an overview of each effort developed, transitioning to reviewing the pros and cons of the methods and discussing problems left open at the end of the review. The chapter will conclude with a reference to the prior work conducted and provide a transition into the work done in this assignment to come up with a wholesome solution that addresses the problems mentioned in the already developed research approaches.

2.1: Review of Existing Literature

This section will be split into three subsections focusing on the following topics and addressing recent literature on each topic: GLOSA/Signal Countdown Timers, Vehicle Countdown Timers, and Traffic Light Object Recognition. The first section will present GLOSA and the concept of using signal countdown timers to prevent unnecessary idling at traffic signals and improve fuel economy of vehicles.

2.1.1: GLOSA/Signal Countdown Timers

Every year, Americans spend over $2,000 dollars in filling up their gas tanks. While this can be partially attributed to spending commute time in congested highways, a lot of fuel is primarily wasted during idling at traffic lights. Multiple solutions to prevent vehicle idling have been designed over the past few years, making use of traffic signal schedules in some way. [6], [8], [10], [11], [21], and [25] discuss developing a GLOSA algorithm that uses short range radar

and real-time traffic signal duration information (V2I communications) to create an optimal velocity trajectory for vehicles approaching the light. Figure 3 represents a schematic of the GLOSA approach.

Figure 3 - Green Light Optimized Speed Advisory (GLOSA)

[14], [15], and [28] discuss the SPAT concept, which uses historical signal timing data combined with real-time updates from the traffic light to project an optimal velocity trajectory for the driver. [22] and [23] refer to using a collection of mobile phones to gather traffic signal timing data. In [22], SignalGuru, a software service that collects this data real-time through an image recognition algorithm that can identify current traffic light states, is introduced. SignalGuru uses DSRC to broadcast and merge the observed traffic light state data with other mobile phones in the communication range. This allows it to understand signal phase timing history for a particular light and derive traffic signal duration predictions as well as enable GLOSA feedback to the driver based on these timings. Figure 4 shows the operation of Signal Guru on live stream video capture.

Figure 4 - SignalGuru App Deployed on Live Stream Video Capture

[23] discusses a general traffic light assistant which receives already existing data from a central server maintaining traffic light timing information. The assistant uses the data received from the server for GLOSA applications and other usages. [39] delves into using real-time signal timing information to provide drivers advice on gear choice or shifting instead. The idea behind these approaches are to adjust the speed of the vehicle to allow for a smooth and safe encounter with the traffic light, either cruising through it (if in the green state) or slowing down to a stop and turning off the engine (if in the red state). While these studies delve into the impact of idling on vehicles in America, [31] and [34] represent the severity of the same situation in more traffic congested parts of India, Madhya Pradesh and Delhi respectively. [31] reveals that about 3750 liters of fuel are being consumed solely due to idling every day in Indore, which is the largest city in Madhya Pradesh. [34] shows even more alarming statistics in Delhi, with over 0.41

million liters of fuel wasted due to idling at the 600+ signalized intersections in the big city. Both sources suggest the use of signal countdown timers, installed on the traffic light to provide drivers an idea of how long the current signal phase will last. Figure 5 shows an implementation of a signal countdown timer.

Figure 5 - Signal Countdown Timer

Countdown timers, in addition to strict idling prevention laws enforced by the government, will coerce the driver to turn of the engine while waiting at a traffic light during the red phase, or adopt a higher speed to travel through the light while it is still in its green phase.

2.1.2: Vehicle Countdown Timers

Section 2.1.1 concludes with a proper transition into one of the main efforts developed to predict and utilize traffic signal timings: vehicle countdown timers. Many automobile companies, such as Audi, Cadillac, BMW, etc. are interested in implementing a built-in

countdown timer or app that offers traffic light duration predictions, in their vehicles, to keep the driver more informed on the road. Audi has already seen some of the fruits of these efforts in big cities across the country, including parts of Oregon, northern Virginia, Las Vegas, and Washington DC, etc. by releasing newer models of their vehicles with the Audi Connect Traffic Light Information System to the mass public, detailed in [12], [16], [17], [18], [19], and [2]. This is the first V2I technology implementation in automobiles that uses a cellular connection to obtain real-time updates from the traffic light, to predict signal durations and simultaneously display them to the driver. The on-board diagnostics showcasing this information is named the Audi Red Light Countdown Clock; and it is structured to provide a visual of the current state of the traffic light ahead of the driver, with the amount of time left for the state to change displayed underneath the visual (updated every second), as shown in Figure 6. Audi has established contact with Traffic Technology Services (TTS) in multiple states to obtain these traffic light timing updates, for fixed and adaptive signals alike.

Figure 6 - Audi Red Light Countdown Clock

BMW is currently pursuing more efforts in this area as well, by incorporating the EnLighten app, discussed in [30], into their future model vehicles. EnLighten uses machine learning and predictive analysis to determine signal length predictions from a large set of traffic data offered to it by its host company, Connected Signals. The data acquired by Connected Signals is fit into multiple learning models that include the current traffic flow effect in determining signal length predictions. The model that provides the most accurate prediction is used by the app to update the driver on traffic light behavior real-time. Figure 7 shows the EnLighten App in operation inside a BMW vehicle. The structure of the app is very similar to the Audi Red Light Countdown Clock.

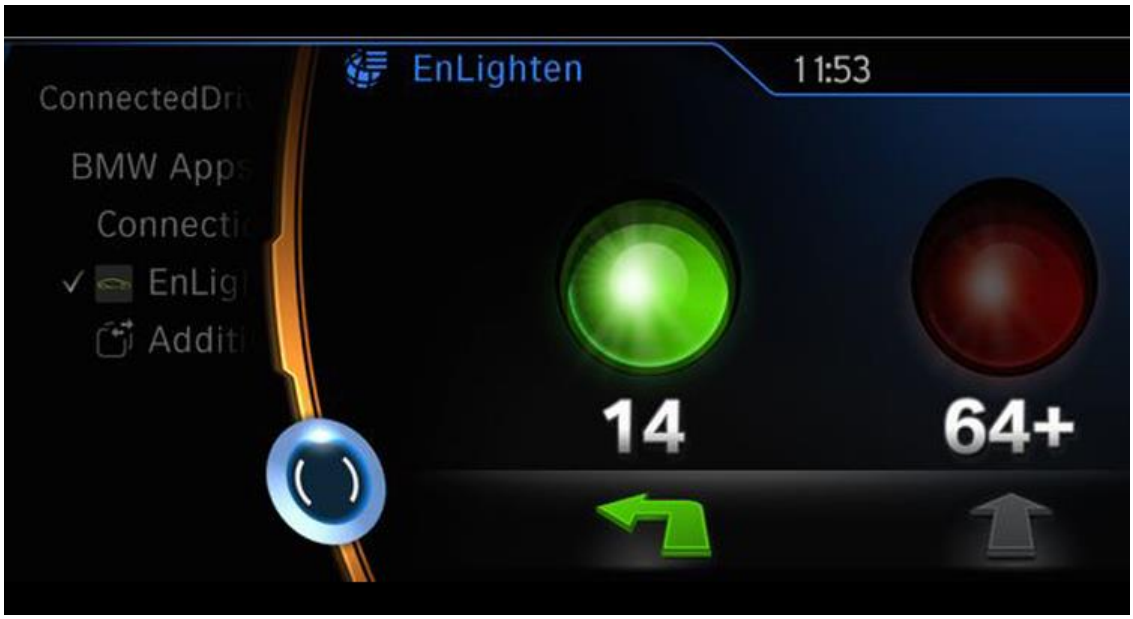

Figure 7 - EnLighten App Connected Inside BMW Vehicle

Cadillac is also starting to look into V2I implementation in its vehicles, and has run some traffic signal prediction tests near the General Motors Warren Technical Center campus, talked about in [42]. The technology developed in this effort currently only supports the state of Michigan, relying on data from the Michigan Department of Transportation.

While there are other lesser-known approaches in determining traffic signal durations and offering predictions back to the driver, the solutions discussed above are the most widely-used and credited traffic light learning and prediction systems, and hence will be the only ones presented in this review.

2.1.3: Traffic Light Object Recognition

Object recognition has turned into an emerging technology in autonomous vehicles, where collision avoidance and advanced driver assistance systems are becoming crucial to the success of vehicles operating without user input. The human eye can perceive and process multiple objects in a frame at once, allowing for drivers to make informative decisions in a jiffy on the road. Object recognition models are striving to achieve performance that can mimic the human eye as closely as possible (it is impossible to get the same level of detection). To attain this, models must be trained on a large set of similar images using a neural network framework, until losses diminish and stay constant at a small value. Training is performed one iteration at a time, where an iteration consists of a forward and backward propagation through the layers of nodes in the network (input, hidden, and output). Figure 8 shows an example neural network, with calculations to find the output node values (forward propagation) and update the weights for a new iteration (backward propagation) based on the error between the actual values and the output values found.

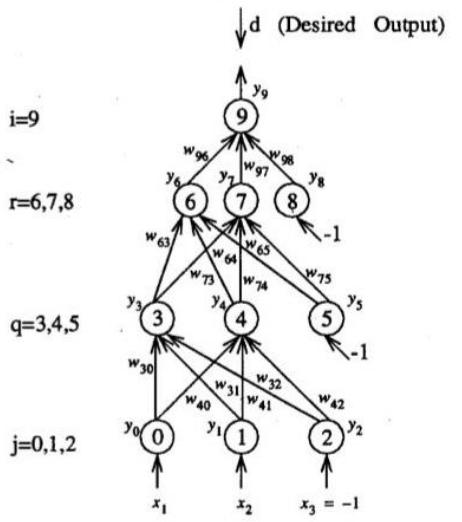

Figure 10.8 Back-propagation network in Example 10.5.

For hidden layers, from Eq. (10.44) we have the following error signals:

$$\delta_6 = a_6'(\text{net}_6)\sum_{i=9}^{9} w_{i6}\delta_i = y_6(1-y_6)w_{96}\delta_9,$$

$$\delta_7 = a_7'(\text{net}_7)\sum_{i=9}^{9} w_{i7}\delta_i = y_7(1-y_7)w_{97}\delta_9,$$

$$\delta_3 = a_3'(\text{net}_3)\sum_{r=6}^{7} w_{r3}\delta_r = y_3(1-y_3)(w_{63}\delta_6 + w_{73}\delta_7),$$

$$\delta_4 = a_4'(\text{net}_4)\sum_{r=6}^{7} w_{r4}\delta_r = y_4(1-y_4)(w_{64}\delta_6 + w_{74}\delta_7).$$

Then the weight update rules are [see Eq. (10.43)]

$$\Delta w_{96} = \eta\delta_9 y_6, \quad \Delta w_{97} = \eta\delta_9 y_7, \quad \Delta w_{98} = \eta\delta_9 y_8 = -\eta\delta_9,$$
$$\Delta w_{63} = \eta\delta_6 y_3, \quad \Delta w_{64} = \eta\delta_6 y_4, \quad \Delta w_{65} = \eta\delta_6 y_5 = -\eta\delta_6,$$
$$\Delta w_{73} = \eta\delta_7 y_3, \quad \Delta w_{74} = \eta\delta_7 y_4, \quad \Delta w_{75} = \eta\delta_7 y_5 = -\eta\delta_7,$$
$$\Delta w_{30} = \eta\delta_3 y_0, \quad \Delta w_{31} = \eta\delta_3 y_1, \quad \Delta w_{32} = \eta\delta_3 y_2 = -\eta\delta_3,$$
$$\Delta w_{40} = \eta\delta_4 y_0, \quad \Delta w_{41} = \eta\delta_4 y_1, \quad \Delta w_{42} = \eta\delta_4 y_2 = -\eta\delta_4.$$

Figure 8 - Example Neural Network with Output Node Values and Weight Updates (Calculated)

Nowadays, all vehicles with a computer vision system incorporate the capability to detect traffic lights of various configurations. It is extremely useful and important to determine the state of a traffic light in order to make a decision on whether a driver can pass through the light (green) or must slow down to stop at the light (red). As mentioned earlier in this chapter, this can prevent fuel wastage due to idling, which is a concern that automobile companies are striving to eliminate.

Deep Learning Frameworks have been developed to implement artificial neural networks with ease. [33] and [37] discuss the TensorFlow Object Detection API, a deep learning framework developed by Google to support models built on existing artificial neural networks or new ANNs. A few of the most popular ANNs used for object detection that are supported by this API are Faster R-CNN, SSD, and YOLO. [36] talks about using Faster R-CNN for more accuracy in detecting smaller objects, such as traffic lights, as opposed to SSD and YOLO, which are both better at identifying larger objects and train in a shorter amount of time compared to Faster R-CNN. [32] sets forth the opposite argument, claiming that by performing detection in earlier and later layers of the neural network (as opposed to detecting in just one later layer) and replacing the original ANN with an Inception network, the SSD can be used to train high performing traffic light recognition models at a faster rate and with more accuracy than Faster R-CNN and YOLO.

There are additionally various detection solutions that have been developed to perform traffic light recognition. [4] discusses results from the Kettering University SAE challenge for developing an autonomous vehicle, presenting the most common detection methods developed and the final solution they arrived at for implementing inside their vehicle. The article talks about the pros and cons of color thresholding, blob analysis, morphological filtering, spot light detection, watershed, etc., finalizing on a solution of performing spot light detection first to identify bright spots, and then doing a watershed transformation to blend the spot lights and determine the objects that remain the same size (representing the traffic lights). Spot light detection and watershed transformation are represented in Figures 9 and 10 below.

Figure 9 - Spot Light Detection Performed on a Signal Image to Identify Bright Regions

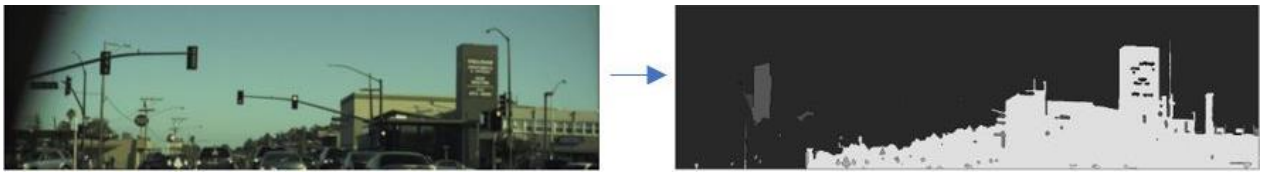

Figure 10 - Watershed Transformation Performed to Blend Spot Lights

[41] presents a different view, highlighting the advantages of blob analysis over other detection methods. blob analysis involves filtering dark connected regions of an image to determine the boundaries of the foreground object, which represents the traffic light in this case.

Last but certainly not least by importance, multiple labelling methods to detect traffic lights on live stream video and images have been developed in the recent research work for this field. [5] and [9] identify traffic lights by detecting the entire box containing the light, allowing the model to distinguish it from a car's tail-lights or other lights on the road. This methodology is shown below in Figure 11.

Figure 11 - Labelling Method 1 – Entire Traffic Light Box

[40] adopts a slightly different approach, detecting just the colored circle as the traffic light configuration instead of the entire box. Figure 12 represents this method.

Figure 12 - Labelling Method 2 – Circle Representing Traffic Light Color

The usability of both of these methods will be discussed in the next section, which details the pros and cons of the above-mentioned solutions.

2.1.4: Pros and Cons of the Approaches Presented in 2.1.1 – 2.1.3

Developing an optimal velocity trajectory to guide drivers through an intersection requires analyzing a lot of data to make calculations. For example, [6] talks about determining a reference velocity by creating an algorithm incorporating the driver's current cruise speed and distance to the traffic light. This a good way to proportionally increase or decrease speed based on distance feedback without exerting too much force on the brakes, preserving vehicle part lifetime. The main drawbacks to the methodology discussed in [6] are due to the vague nature of assumptions made by the model. Since real traffic light timing data was not available for the simulation conducted, the dissertation discusses using an arbitrarily determined traffic light schedule, the rationale and methodology behind which was not discussed. Further, signal timings

are assumed to operate on a fixed schedule which additionally does not incorporate the signal phasing effect that could be introduced at various times of the day due to traffic flow.

The functionality of the traffic signal countdown timers, discussed in [30] and [34], while extremely useful in determining accurate signal length predictions and relaying them back to the driver, fall short in a few other respects. Traffic signal countdown timers are expensive additions to the existing infrastructure, especially if they need to be installed at all intersection points. Further, these timers are not highly accurate; generally, traffic lights tend to change state a few seconds after the timer predicts it, which can lead to drivers starting their vehicles prior to the actual state change, eventually building up fuel wastage concerns again.

Vehicle countdown timers discussed in [12], [16], [17], [18], [19], [30], [2], and [42] are more useful on the other hand since they are built into a vehicle or offered as an application that drivers can use on their phone. This is much less expensive compared to making changes to the original road infrastructure and even has the advantages of offering faster, real-time signal duration updates to drivers due to cellular connectivity and DSRC technology implemented in vehicles. A major drawback to vehicle countdown timers is their heavy reliance on companies, such as TTS, to obtain state traffic light timing data. While this data may be easily provided in some states with connected signals and infrastructure, other states would require extensive approval and procedures to obtain this data, especially if there are no independent providers and all traffic timings are mandated by the government-controlled traffic departments. In the situation just described, problems with how the traffic signal timing data is organized may also arise, making it more difficult to form and relay signal length predictions back to the driver.

Creating a traffic light recognition model using the TensorFlow Object Detection API Deep Learning Framework is a smart way to reduce the efforts involved in developing a platform to host a neural network structure. With TensorFlow, object recognition models can be developed easily by gathering and labelling images, converting the labelled annotations to TensorFlow record files, and submitting them to an already established neural network structure such as SSD, Faster R-CNN, etc. to train the model, as explain in [33] and [37]. Further, as discussed in [32] and [36], the neural network implementation chosen to train the model depends on the detection priorities; if the user is looking to develop a model that can be trained fast with reasonable accuracy for small and large objects, the original SSD network structure is the way to go for them. Users with more time on their hands and wanting to build a highly accurate model for smaller object detection should use the Faster R-CNN network structure. Distinguishing traffic lights from their background and other similar images can be done by specifying a proper detection technique. [4] and [41] discuss detection methods that segment images by blobs of pixels that are the same color, which is useful in determining a mass of pixels corresponding to a particular object. However, this does not qualify the object to be a traffic light, especially on the road where multiple objects have similar orientations and pixel densities. Color thresholding and overall shape detection (through labelling), should be added as additional factors in narrowing down a detection to a traffic light as opposed to another similar looking object.

2.2: Prior Work – Focus on Improving Fuel Economy and Energy Efficiency of Vehicles

The work done in this research can be split into four phases, as represented below in Figure 13.

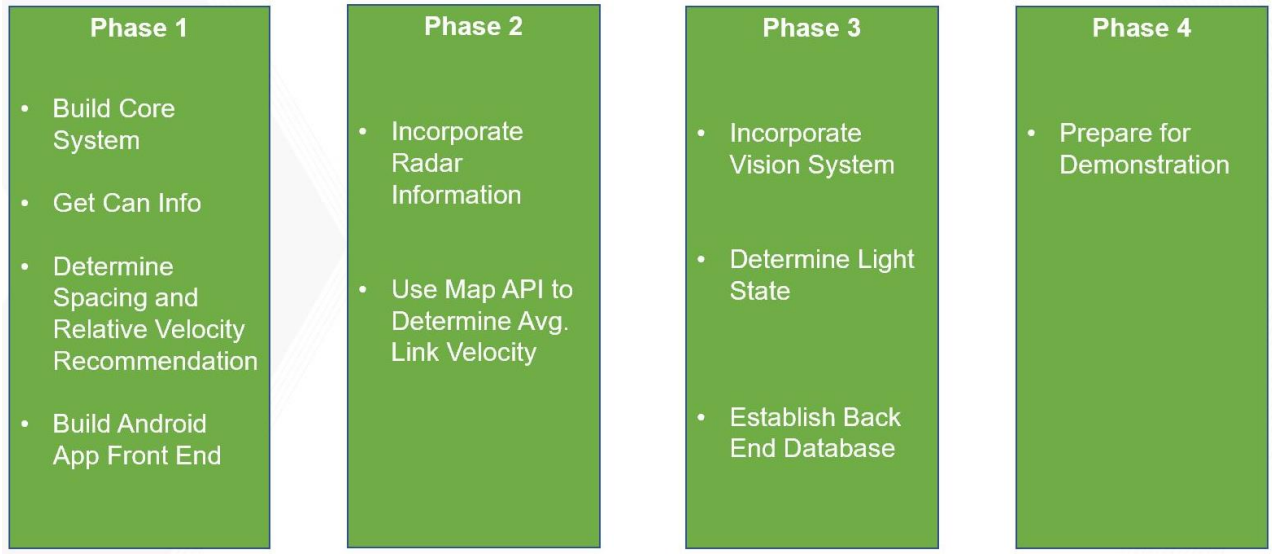

Figure 13 - Chart Representing the Four Phases of Project Work

The prior work to the current research assignment fulfills Phases 1 and 2. The goal of this work is to address fuel economy and energy efficiency concerns in vehicles by preventing idling at traffic lights. The work creates two Android apps as a part of this process; one to learn the phase durations of traffic lights on a set route at a certain defined time period and another to coach the user on optimizing their speeds through the light and maintaining space with respect to the vehicle in front of them based on radar feedback. These are called the Eco-Learning and Eco-Driving apps respectively. In order to learn the duration of the signal, the prior work suggests manual uploads by a tester into the learning app upon observing phase transitions at each light. The signal durations are assumed to operate according to a fixed two-phase schedule, for example, a 30 second green and 1-minute red phase alternating eachother. As the driver encounters the traffic light, the tester reads the signal durations out to them, so that they can adjust their speed accordingly to either make the light on time, or slow down to a stop. Figure 14 shows the display of the Eco-Driving app.

Figure 14 - Eco-Driving App Display (Prior Work)

The rest of this document will focus on how the current work fulfills Phases 3 and 4; outlining the methods developed to create a real-time traffic light learning and prediction system and detailing the results obtained from testing this system.

CHAPTER 3. METHODS EMPLOYED TO DEVELOP TRAFFIC LIGHT LEARNING & PREDICTION SYSTEM

This chapter starts off by providing a summary of the research methods to be discussed. It then delves into an overview of the leader/follower approach, specifics of the developed systems for traffic light learning and prediction, and a briefing of the driver display unit. First, the design of each sub-module and its role in the overall system is discussed; later the big picture of the design of the integrated system and its workings is provided. Figure 15 shows a flowchart that represents the functions of the integrated system and how they interconnect with one another. This chart will be referenced and discussed in section 3.5.

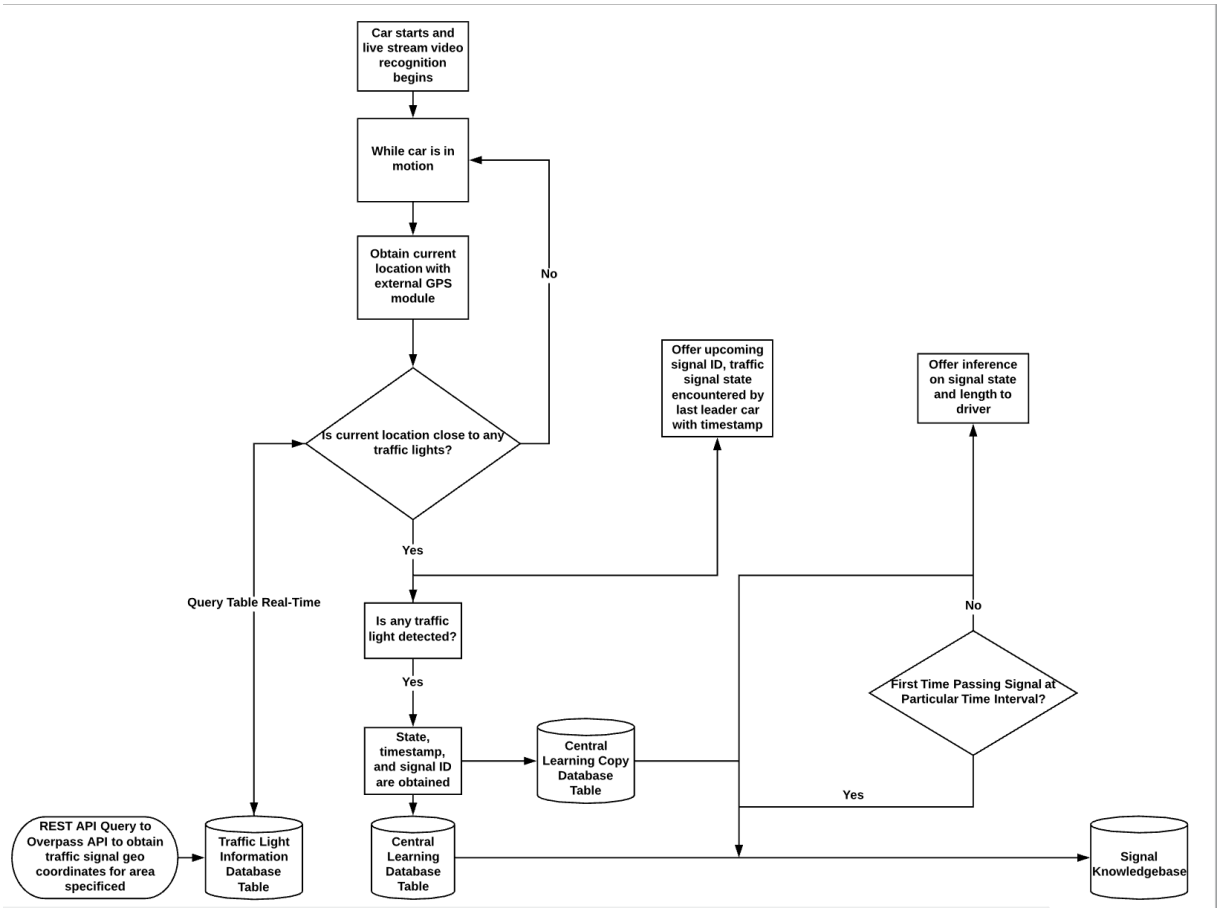

Figure 15 - Flowchart of the Integrated System

3.1: Summary of the Research Methodology

Two main algorithms, one to accomplish traffic light learning, and the other to accomplish traffic light prediction, were developed and will be explained in this section. The traffic light learning algorithm involves using a computer vision system to detect and process traffic lights captured in live stream video while driving on the road. The traffic light prediction algorithm uses all of the processed light detections to determine when the state changes occurred and applies machine learning techniques to develop and refine signal length predictions. These predictions are relayed back to the driver to inform them of the approximate signal change time of the traffic lights ahead of them, so they can plan ahead to cruise through the green phase (while staying under the state mandated speed limit) or at least turn off their engine while waiting at a red phase. Another method that was incorporated in designing both the learning and prediction algorithms was the leader-follower approach, discussed in section 3.2 of the document. As a summary, this approach categorizes a few vehicles as leaders, due to their ability to "learn" traffic lights and contribute to creating or refining signal length predictions for various traffic signals. The remaining vehicles are categorized as followers since they do not have these capabilities; they will just be able to receive information about signals on their route and adjust their driving plan accordingly. The remainder of this chapter will provide explicit detail on the design of the traffic light learning and prediction algorithms as well as how they incorporate the leader-follower approach.

3.2: Leader/Follower Approach

The leader/follower approach was introduced by the prior research work on this project. The idea behind this approach is to recognize vehicles as a fleet moving together, where one car

serves as the “leader” in providing useful information to the rest of the fleet, and the other cars serve as “followers” by just receiving and processing the information. Any vehicle with an advanced driver assistance system (camera, connection to CLDB) is considered a leader vehicle, since it can upload valuable traffic light timing information after crossing the light. Follower vehicles can pass the same traffic light a few minutes or even hours after the leader vehicle passed it, but they will still be able to access the information uploaded by the leader upon crossing the light. Leader vehicles are also considered to be follower vehicles. This methodology represents that fleets move together but not necessarily at the same speed or time, one vehicle behind another vehicle. Any vehicles following a leader vehicle’s path through the same traffic light is considered to be a follower vehicle.

In the prior work to this research, the leader vehicles do not autonomously upload the traffic light state and time of state observation. This is accomplished by a passenger in the leader vehicle, who manually performs the uploads into an app to determine signal length predictions, which are relayed to other leader and follower vehicles. In this work, a solution to this design problem is introduced to develop an autonomous approach to traffic light learning; a computer vision system for traffic light detection. This allows the leader vehicles to perform the learning process without requiring any manual human labor in uploading the traffic light state detections. The follower vehicles are also able to get updates automatically as they pass through a traffic light, since they can access all of the information from the CLDB and KB. Figures 16 and 17 show schematics of the revised leader-follower approach presented in this document.

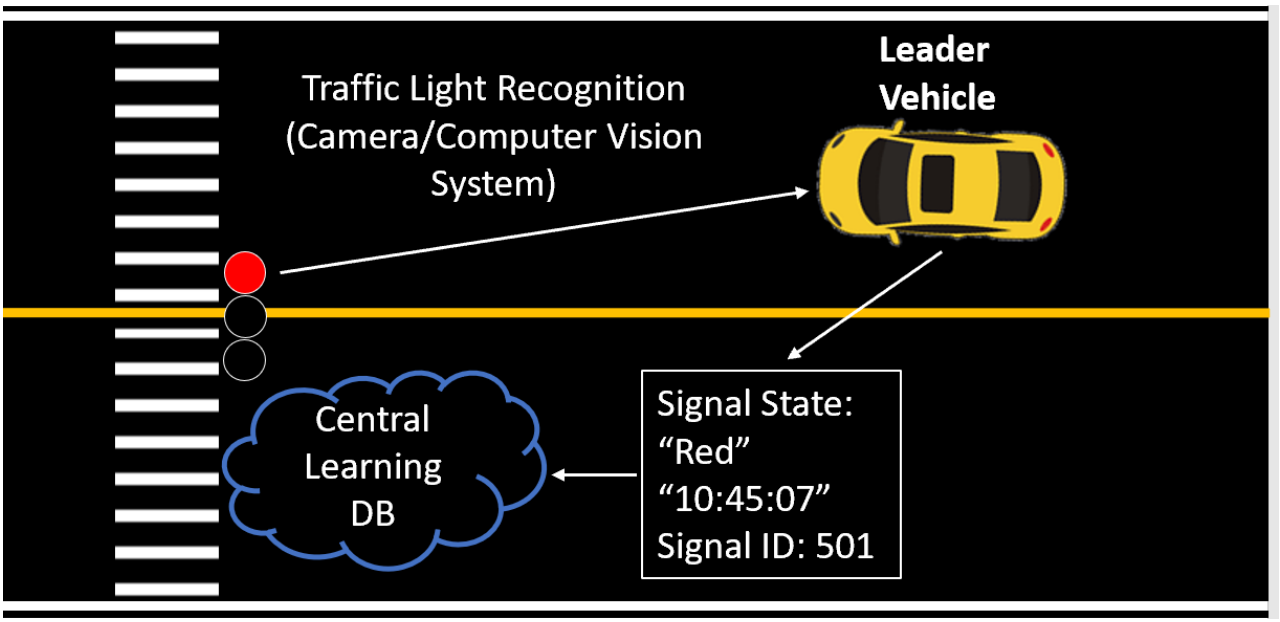

Figure 16 - Schematic of Leader Vehicle Using Computer Vision System to Detect Traffic Lights and Performing Uploads to CLDB

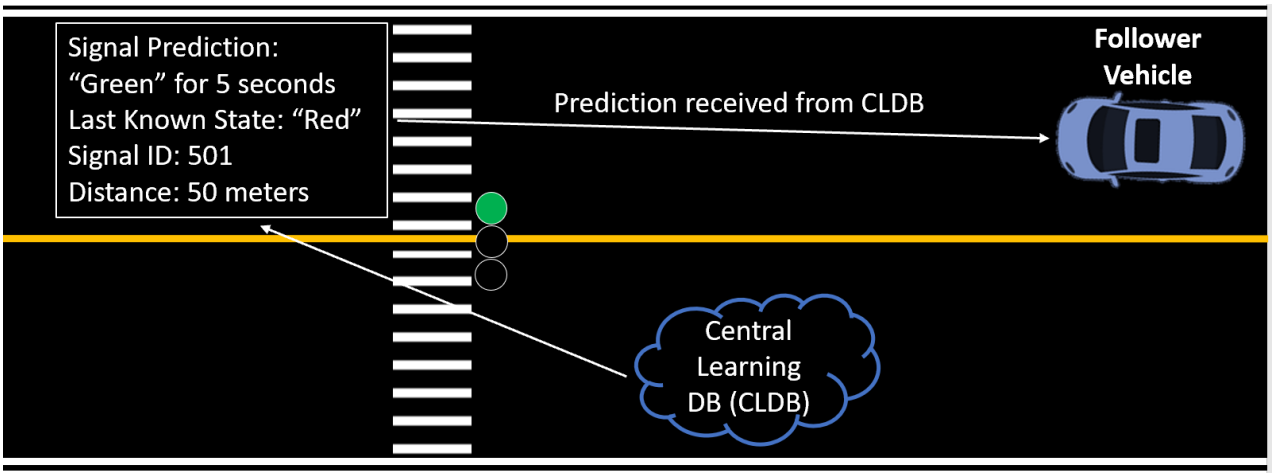

Figure 17 - Schematic of Follower Vehicle Receiving Prediction from CLDB

The next section will discuss the traffic light learning system developed in this work.

3.3: Traffic Light Learning System

The traffic light learning system is used to recognize the traffic signal state in order to process predictions on a signal's phase durations. Figure 18 represents a process flow schematic for the traffic light learning system.

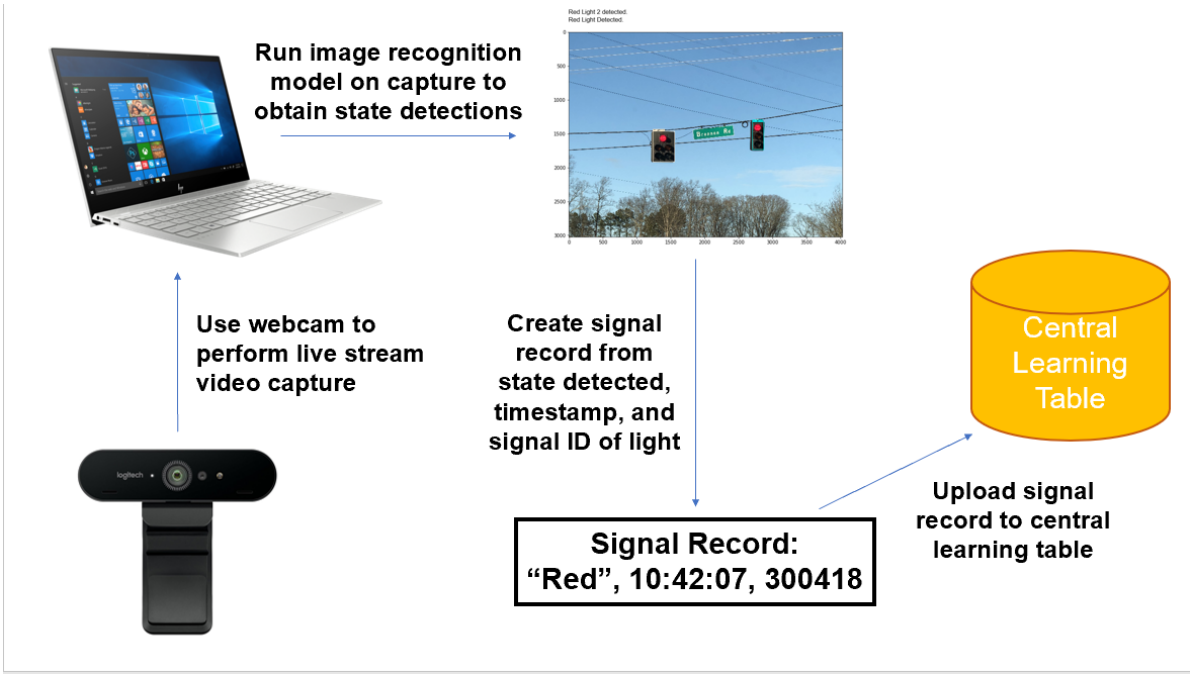

Figure 18 - Process Flow Schematic for Traffic Light Learning System

The learning system utilizes a machine vision algorithm, developed with the Google TensorFlow API for Python framework and SSD Mobilenet V1 Model Configuration File (ANN). The system is trained on approximately 2500 meticulously captured images of traffic lights in Atlanta and the northern Georgia area, and additionally uses Microsoft's COCO pre-trained dataset's traffic light image category to refine image detections with high accuracy. A total of three traffic light recognition models were trained, with slight modifications in labelling and added configurations from one model to another. The second and third model developed in this assignment identified various configurations of traffic lights the best and will be discussed in the following subsections. Details about the first model will be provided in the appendix.

3.3.1: Traffic Light Recognition Model 2: Detection Combining Color-Based Differentiation and Traffic Light Shape

The second model was designed based on autonomous vehicle vision systems. These systems perform detection by identifying the shape of the traffic light along with the color it is displaying. Figure 19 represents example traffic light state detections performed on live stream video by the model in Cumming, Georgia. The success of this model can be attributed to the fact that it can differentiate between a traffic light and similar objects in the background, such as a car's tail lights, for example. This is due to its understanding of traffic light geometry and the spatial location of each color on the light with respect to the surrounding environment.

Figure 19 - Model 2: Green Light State Detections Performed on Live Stream Video in Cumming, GA

A total of 7 different traffic light configurations were submitted to the model 2 training process. This included the standard red and green light configurations, addressed as Red Light

and Green Light respectively, the Red Light 2 and Green Light 2 configurations, and the Red Left, Green Left, and Green Right arrow configurations. All traffic light types trained under these 7 configurations are shown in Table 11 in Section C.1 of the Appendix. The model was trained with approximately 1100 images of traffic lights in Cumming, Georgia and 150 images of traffic lights in the metro Atlanta area. Training was performed for 7 weeks with model loss graphs checked continually using TensorBoard (a loss visualization software developed by TensorFlow) each week. The final model was exported when the losses fell under 2 and stayed stable at that value, indicating that the model was sufficiently trained. Figure 20 represents the final loss graph for model 2. The results obtained using Model 2 for live road testing will be discussed in Chapter 4.

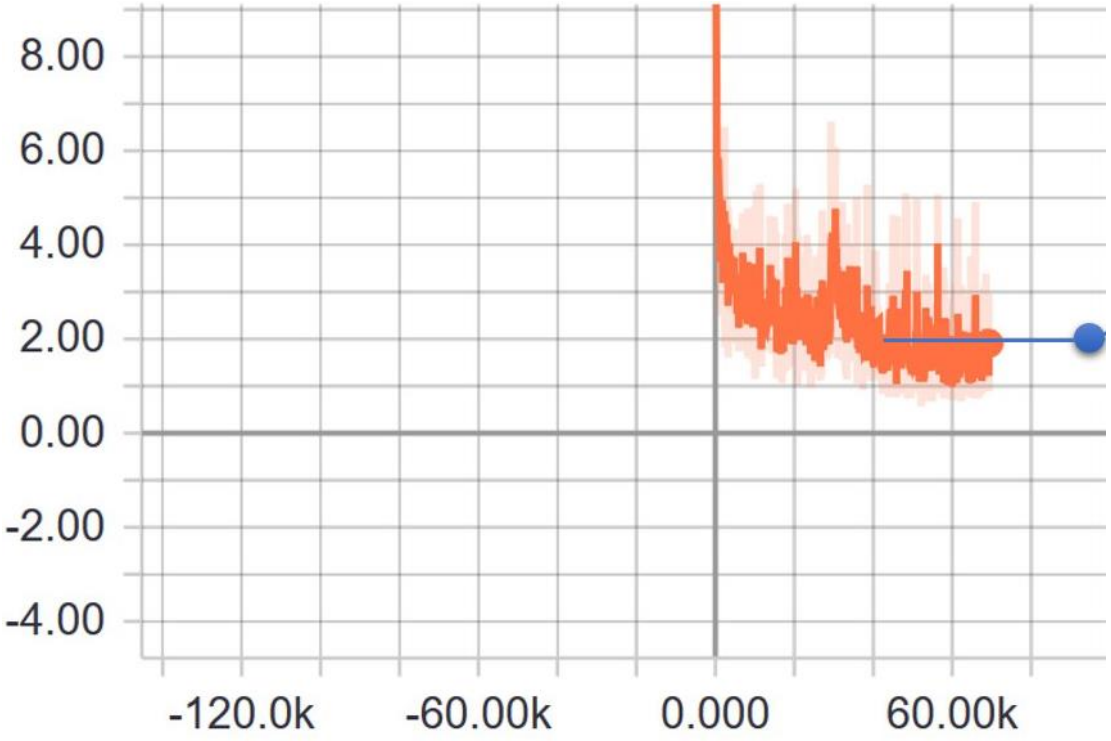

Figure 20 - Model 2: Final Loss Graph Exported After 70k Steps of Training

3.3.2: Traffic Light Recognition Model 3: Detection Combining Color-Based Differentiation and Traffic Light Shape (Focusing More on Atlanta Traffic Lights)

The third traffic light recognition model was built using the second model as a basis, but with a higher concentration on incorporating traffic lights from Atlanta. As shown in the results and discussion section of the document (Chapter 4), Model 2 had high accuracy in detecting traffic light configurations it was trained to recognize in Cumming, Georgia, but failed to produce an equal level of accuracy in detecting the same traffic light configurations in Atlanta. After investigating more into the cause behind this phenomenon, it was determined that the city setting (busy environmental surroundings, tall buildings, etc.) was making it hard for the model to distinguish traffic lights amongst their environment. Model 3 was developed to provide a fix to this problem, by incorporating over 1,300 more images, focusing entirely on traffic lights in Atlanta, to the already collected set of images used to train Model 2. Its aim was to capture and factor in the city setting to the traffic light recognition model framework already developed in Model 2.

Some amends were made to Model 3 before training was initiated. The Red Left and Green Left arrow configurations, shown in Table 11, Section C.1 of the Appendix, were eliminated as they were not well identified during Model 2 live road testing as discussed in Chapter 4. Further, two new configurations, Green Light 3 and Red Light 3, shown in Table 12, Section C.2 of the Appendix, were added to the model due to their prevalence in Atlanta. Training was conducted for 5 weeks and the model was exported when losses reached below 2 and stayed relatively stable. Figure 21 shows the final loss graph for Model 3. The model was trained for a total of 91,000 steps, which includes Model 1 and Model 2 training. Example traffic

light state detections performed on live stream video by the model in Cumming and Atlanta are shown in Figures 22 and 23 respectively. The results obtained using Model 3 for live road testing will be discussed in Chapter 4.

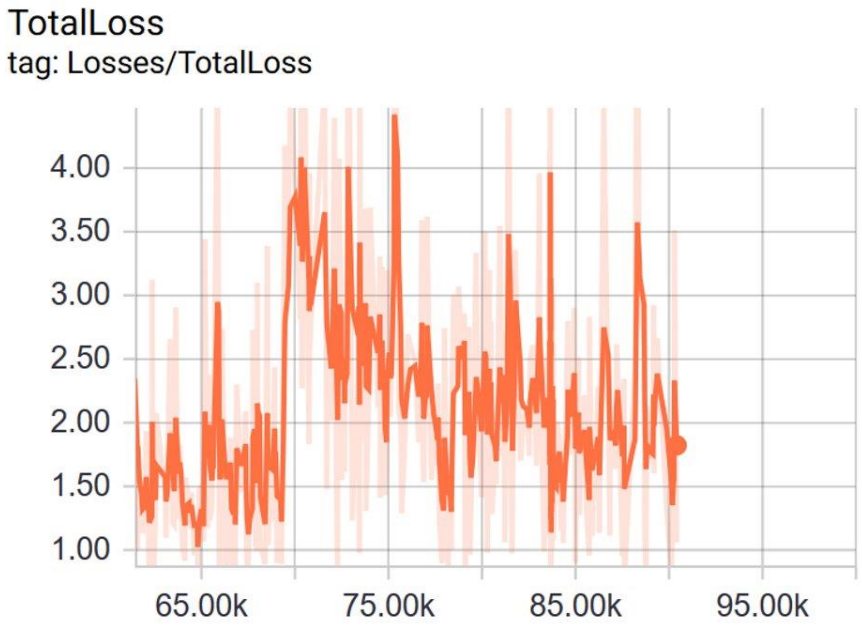

Figure 21 - Model 3: Final Loss Graph Exported After 91k (Total) Steps of Training

Figure 22 - Model 3: Green Light State Detections Performed on Live Stream Video in Cumming, GA

Figure 23 - Model 3: Red Light State Detections Performed on Live Stream Video in Atlanta, GA

The following subsection will provide details on how live stream video recognition was performed during the final test.

3.3.3: Live Stream Video Recognition

Live stream video recognition was implemented in order to perform continuous traffic light detection throughout the drive. This allows for the detected traffic light state to be immediately uploaded to the CLDB along with the timestamp of detection and the signal ID of the traffic light. In order to perform live stream video recognition, a basic video capture and model processing script, developed by TensorFlow, was altered to utilize the final model created (Model 3). Although this script has the capability to recognize objects trained by a model submitted to it on live stream video capture, it cannot refine and save the detections found. To introduce these capabilities in addition to other major ones, the visualization_utils.py script, provided with the TensorFlow model configuration files, was heavily manipulated. An overview

of the functionality of this script as well as the changes introduced to it to fulfill the above-mentioned goals will be discussed in the next subdivision. The overall modified script will also be presented for clarity.

The visualization_utils.py script is provided along with the Google TensorFlow API Object Recognition Model Framework scripts. Its main functions are to process the object recognition model on a picture or video frame, identify the class detections (according to the classes trained by the model), and draw bounding boxes on the original image or video frame to represent the detection to a model user. In this research assignment, the visualization_utils.py script sub function, visualize boxes and labels on image array, is heavily altered to perform the necessary functionalities of the integrated system, explained through the following example.

Figure 24 displays a frozen live stream video frame captured during a sample test run. The frame shows two traffic light boxes representing the green phase, as the driver approaches the signal. When signal records are processed into the CLDB, two green light records for the same timestamp and frozen video frame should not be uploaded into the table, since that is repetitive and unnecessary in creating useful learning data. Rather, the algorithm should be able to narrow down the number of detections found on the frozen video frame to the predominant color of the light (in this case, “Green”), store the found traffic light state in a global variable that can be used in the main system execution code, and finally upload it along with the timestamp at which it was observed and the signal ID of the traffic light into the CLDB when reaching a particular distance threshold of the light. This allows only unique signal records to be uploaded into the CLDB. On the other hand, if the traffic light as a whole was found multiple times by the

model in different video frames, developing separate signal records per frame at different timestamps would qualify as unique signal records to be uploaded into the CLDB.

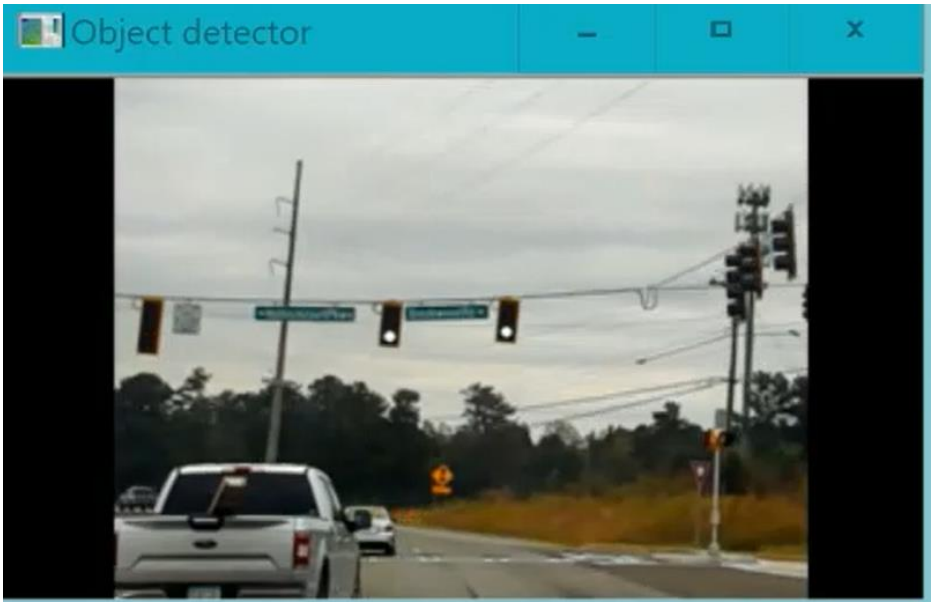

Figure 24 - Frozen Live Stream Video Capture Frame

Every video frame that is processed by the visualization_utils.py script with the model developed will most probably contain more than one traffic light detection. This is due to the fact that it is a rare occurrence to find intersections with just one traffic light. Due to the widespread prevalence of this phenomenon, it is necessary to gather all of the individual traffic light configuration detections found in the frame and store them as state detections in an array. The array is initialized at the top of the visualize boxes and labels on image array function and appended to within certain conditional statements. Once all of the configurations detected in the image have been appended to the array as shown above, another conditional statement checking whether all of the appended detections are of a particular state (“Green” or “Red”) is executed and the final traffic light state detection is stored in a global variable. This variable is initialized every time a new video frame is processed (as is the array), so that no state detections will be

stored in memory. Figure 25 shows the detections corresponding to the frozen video capture frame displayed in Figure 24 and how they are narrowed down to a particular traffic light state.

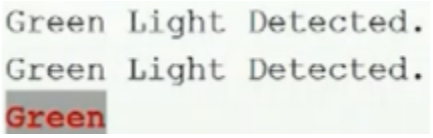

Figure 25 - Frozen Video Capture Frame Detections and Traffic Light State Determination

In order to prevent driver distraction, the bounding boxes representing the state detection on the original video frame were disabled by commenting out the draw bounding box on image array function call in the visualization_utils.py script. The script was then imported as a package into the integrated system program to be able to use the light state in performing signal record uploads to the CLDB, discussed in the next subdivision of this section.

In order to be able to offer predictions on signal duration to a driver, it is necessary to have learning data on the signal state transitions. To create learning data, signal records consisting of the traffic light state detection, timestamp, and signal ID are uploaded as detected real-time into the central learning and central learning copy database tables, when a driver is approaching the light at a certain distance and in the same directional heading.

The first thing to determine is the driver's current distance to the signal ID and compare it with a threshold to ensure that the traffic light recognition model has identified an actual light within proximity of the current location. Sometimes, the model may misidentify objects as traffic lights due to poor weather or lighting conditions, as discussed in Chapter 4 which presents the results of the model performances; hence it is important to check the distance condition first. The next thing is to determine whether a valid traffic light state detection was found at the time of entering within the threshold distance. In the case of this model, this would be a red or green

light detection (even green right arrows qualify as green lights to simplify the scenario) found within the threshold distance. After this has been verified, it is important to ensure that the direction in which the driver is travelling is the same as the direction in which the traffic light is facing. The manner in which the directional heading was determined is explicitly detailed in section 3.4.1. If the directions match, a signal record upload can be warranted. A schematic representing these three checkpoints for warranting a signal record upload is presented below in Figure 26.

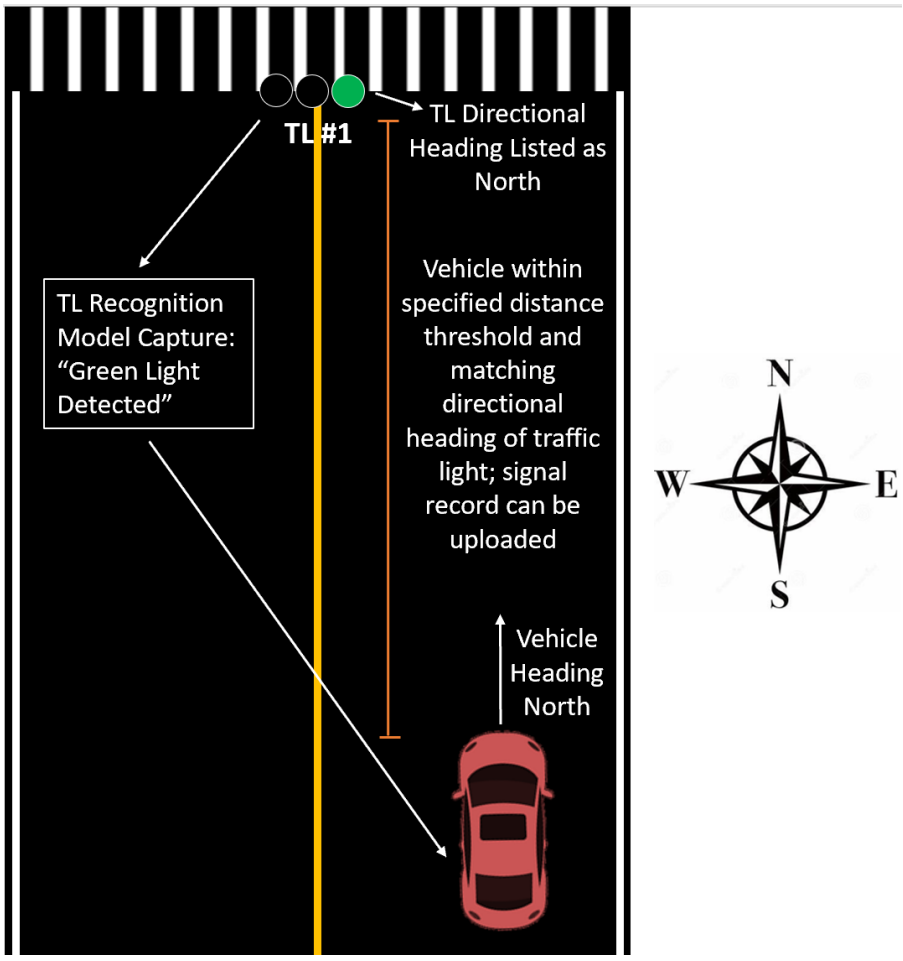

Figure 26 - Three Checkpoints for Warranting a Signal Record Upload (Schematic)

The timestamp of the detection is then found by getting the current time using Python's datetime package and the traffic light state stored in the global variable (obtained from the output of visualization_utils.py as discussed in the previous subdivision) is invoked. Finally, two insert queries (one to the CLDB and one to the CLDB copy) are issued, submitting the timestamp, traffic light state, and signal ID as a record to be uploaded for "learning."

The next section of this document will focus on the second system developed in this work; the traffic light prediction system.

Section 3.4: Traffic Light Prediction System

The Traffic Light Prediction System is intended to inform a driver on the predicted signal lengths of traffic lights on their route, so that they can organize their driving behavior to minimize travel time and fuel economy, in addition to other prominent factors. Additional functions that the system can perform are alerting the driver that a traffic light is upcoming on their route within a particular distance and informing them of the last known state of that particular traffic light (if it exists). Figure 27 represents a process flow schematic for the traffic light prediction system.

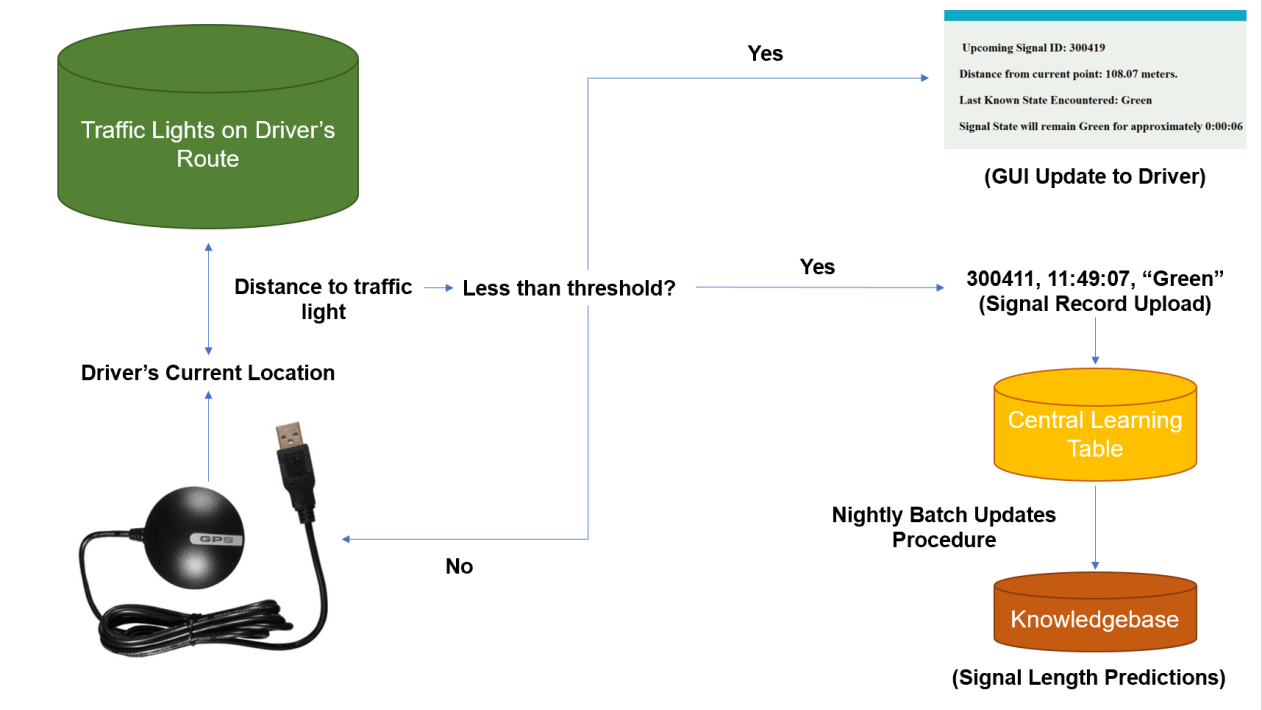

Figure 27 - Process Flow Schematic for Traffic Light Prediction System

The following subsections will focus on the details of developing the modules required to generate these updates.

3.4.1: Finding the Nearest Traffic Lights to a Driver's Current Location

Knowing which traffic light is close to a driver's current location as it continually changes is the basis for this entire prediction algorithm, and hence this is the first sub-module to consider in the whole system. To determine the driver's current location every second with accuracy, the Global Sat BU-353 S4 USB GPS Receiver was used. The GPS Receiver provides the changing latitude and longitude coordinates in the form of serial data that can be decoded and used in computer programs. The device output is highly accurate since it communicates with and receives GPS data from 12 satellites.

In order to use the GPS Receiver, the USB end must be plugged in to a port in the PC. The program executing this function specifies the GPS Receiver to be plugged into the COM 5 port of the user's PC. The user can utilize this port without making any changes to the code provided, or change the port as per their device constraints. The GPS Receiver must be installed in the system using its OS specific driver (for this assignment, a PC running Windows 10 Pro Version 1809 was used and hence the PL2303 Prolific Driver Installer v_1417 was installed as the GPS driver).

All the traffic lights on the driver's route are stored with their signal IDs and geographical coordinates in the Traffic Light Information Database (TLIDB) Table. After decoding the serial data from the GPS Receiver, the module executes a query using the latitude and longitude obtained from this data as an input to find the nearest traffic light to the driver's current location from the table. This query orders the distance between each traffic light's geographical coordinates from the current location's geographical coordinates, using the Haversine Formula for distance calculations. It then selects the matching traffic light that has the smallest distance to the current location and feeds it back to the program. Figure 28 represents the Haversine Formula and Figure 29 shows a flowchart for this process.

$$6371*\cos^{-1}(\cos(\text{radians}(\text{latitude\_1})) *\cos(\text{radians}(\text{latitude\_2})) * \cos(\text{radians}(\text{longitude\_2}) - \text{radians}(\text{longitude\_1})) + \sin(\text{radians}(\text{latitude\_1})) * \sin(\text{radians}(\text{latitude\_2}))) * 1000$$

Figure 28 - Haversine Formula

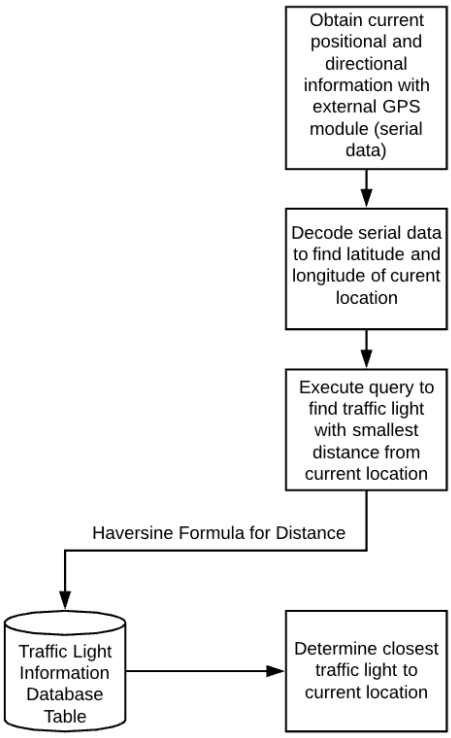

Figure 29 - Finding the Nearest Traffic Lights to a Driver's Current Location (Flowchart)

In order to ensure that signal record uploads are processed for the traffic light truly closest to the driver's current location (in the direction of travel), a formula to calculate heading is executed every time the GPS is polled. The heading obtained from this formula is then compared to the directional headings of the traffic lights found from the closest lights query, shown in the above figure. If the headings match, the signal records will be uploaded upon reaching the threshold distance as explained in section 3.3.3. This approach works best when vehicles are in direct line of sight to the traffic light closest to their current location. The next subdivision discusses the formula for calculating the heading.

The heading is the angle calculated between two points of a driver's current location. For ease, these two points are taken to be an instance of the driver's location at the most recent GPS poll and an instance at the preceding GPS poll. Using the latitude and longitude values of these

points, the formula represented below in Figure 30 can be executed and the x and y values to determine direction can be found. The atan2 function in Python allows for the heading to be computed with respect to all four quadrants. This heading can be converted to degrees by multiplying with a factor of 180/pi.

$$X = \cos(lat) * \sin(lon - prevlon)$$
$$Y = \cos(prevlat) * \sin(lat) - \sin(prevlat) * \cos(lat) * \cos(lon - prevlon)$$
$$Heading\ (degrees) = \tan^{-1}(Y/X) * (180/\varpi)$$

Figure 30 - Formula to Calculate Heading Between Two Points

Using the heading angle found in degrees, the direction of travel can be determined by placing the angle over Figure 31. Tolerances were set for each directional heading at 2 degrees before and after the actual threshold in order to give more leeway in calculating the driver's direction. This is due to the fact that it is hard to always travel true to one of the four compass directions; there will almost always be a little bit of a tilt from the true normal that lies between two directions and it is important to include this tilt when determining the closest light to the driver's current location. Additionally, the headings are calculated counterclockwise rotationally from 0 degrees. The thresholds for each direction with tolerancing have been listed in Table 1.

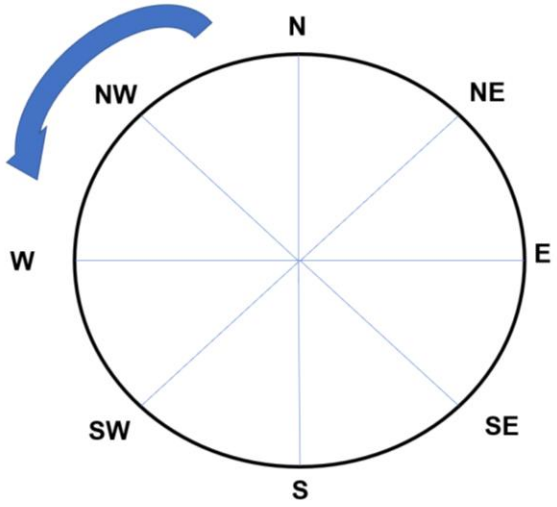

Figure 31 - Schematic of Directional Headings

Table 1 - Directional Thresholds Tolerancing Ranges

$358 \leq degree \leq 2$	N
$2 \leq degree \leq 88$	NW
$88 \leq degree \leq 92$	W
$92 \leq degree \leq 178$	SW
$178 \leq degree \leq 182$	S
$182 \leq degree \leq 268$	SE
$268 \leq degree \leq 272$	E
$272 \leq degree \leq 358$	NE

3.4.2: Upcoming Traffic Light on a Driver's Route

Providing a driver information on the upcoming traffic light on their route is essential to prevent collisions due to distraction or not being able to see the light when driving on curvy roads. As mentioned earlier in this document, all traffic lights used for testing are maintained

with their signal IDs and geographical coordinates in the TLIDB table. When the driver approaches a traffic light within a particular distance threshold, its signal ID as well as distance from the driver's current location will be displayed on a GUI update.

In order to ensure that the traffic light fed back to the program as closest to a driver's current location is truly the closest, a check on whether the distance to the signal is continually decreasing is performed every iteration. This is done by comparing the current distance to the distance noted in the previous iteration. By doing this, incorrectly identified traffic lights at different sides of the same intersection are discarded and only the true upcoming signals on a driver's route are shown on the GUI. If the distance is found to be constantly decreasing with reference to a particular traffic light and is within a certain threshold limit of that light, the program proceeds to create and display the GUI with confidence on the upcoming traffic light on the driver's route. Figure 32 shows a schematic that represents this ideology.

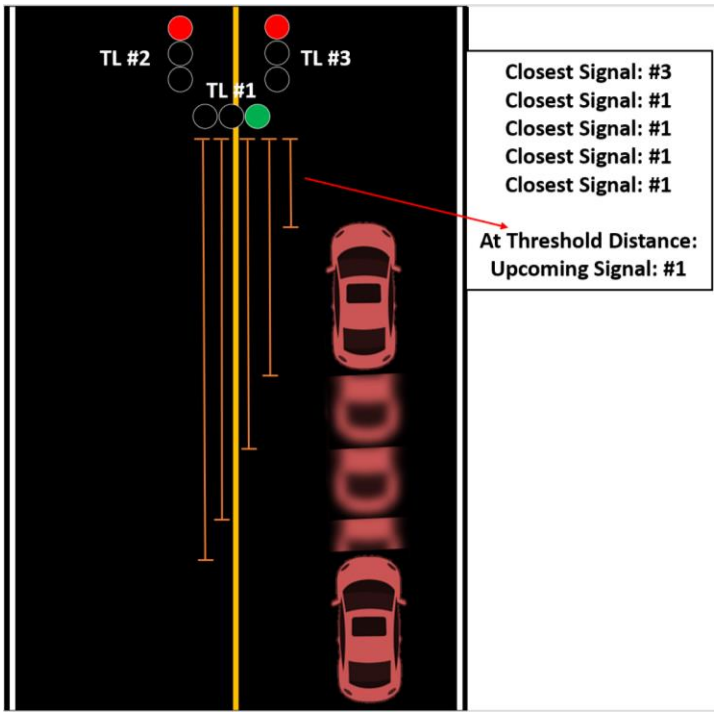

Figure 32 - Determination of Upcoming Traffic Light on a Driver's Route (Schematic)

The next subsection discusses relaying the last known state of the upcoming traffic light to a driver, if it exists.

3.4.3: Last Known State of the Upcoming Traffic Light on a Driver's Route

Traffic lights in very busy downtown or inner-city areas have the ability to offer strong predictions on their state changes, due to the high volume of learning data uploaded by cars passing through them. On the other hand, traffic lights in suburban and rural areas, such as small towns and farms, do not have as much learning data to generate strong signal length predictions, due to the low volume of commuters passing through them. In order to make the traffic light learning and prediction model discussed in this thesis widespread and applicable for commuters coming from all types of areas, the last known state of a traffic light on a driver's route is displayed along with a signal length prediction (if one exists for that particular light during the current time period). The purpose of this information is to make the driver aware of the state of the traffic light when the most recent leader car passed through it, so that they can use it along with the already provided signal length prediction to make an educated guess with reasonable accuracy on when the traffic light state will change. This last known state could potentially be uploaded hours or days ago, but since the assumption in this project is that traffic lights follow a fixed schedule every day of the week, the drivers can still use the information to predict when the state will change.

When the driver approaches the traffic light at a particular distance threshold, the signal ID for the light (found from the Finding the Nearest Traffic Lights Module) is coordinated with the most recent signal record upload for that light from the CLDB Copy, maintaining all historical signal record uploads. This is done by executing a query that orders the records by

timestamp and selects the max record to feed back to the program. An example of this is presented below in Figure 33. If a driver encounters a signal at 8 pm on a current day and records for that particular signal respectively exist at 2, 4, and 6 pm on the same day in the CLDB Copy, the last known state read back to the driver will correspond to the state stored for the 6 pm record, due to it having the maximum timestamp.

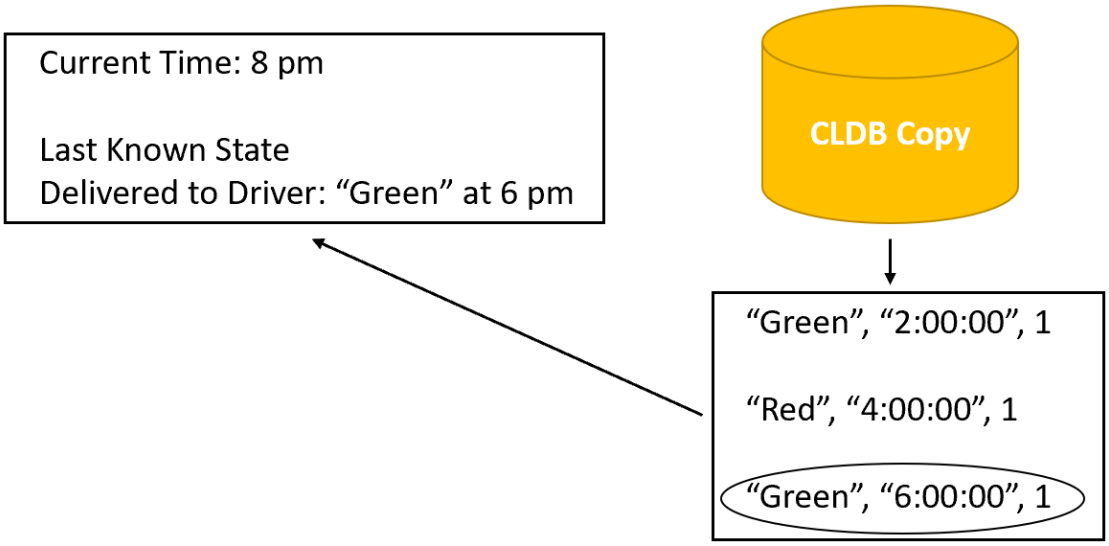

Figure 33 - Determination of Last Known State of a Traffic Light (Schematic)

The record fed back to the program is relayed to the driver on the same GUI update offering information on the upcoming traffic light on a driver’s route, but in a separate text widget. This will be explained in more detail in section 3.6.1, which discusses the structure and implementation of the GUI. Providing the last known state update allows the driver to make more informed decisions about passing or stopping at an intersection, based on historical data provided to them about the traffic light’s phases. The next sections of this document discuss the most important subfunctions of the traffic light prediction system; forming and relaying signal length predictions to the driver.

3.4.4: Signal Length Predictions

This is one of the most important parts of the overall system since it provides the driver with the actual prediction on when the traffic light will change state. This module involves two parts; forming the predictions and relaying the predictions back to the driver. The following subdivisions will discuss how signal length predictions are formed based on learning data.

As mentioned frequently throughout this thesis, a signal record consists of a signal ID, time stamp, and observed traffic light state. The goal is to order all of the signal records for a particular light into signal length predictions by finding same state uploads that last for a certain amount of time and determining where the state changes occurred among all of the records. In order to do this, the NBUSP was run every day on the records maintained in the CLDB to store predictions in the KB. As a reminder, the CLDB is emptied every night after this process is complete, while the CLDB Copy maintains all of the records uploaded to it as historical data, as mentioned in the previous section.

The functions performed by the batch updates procedure are represented in a function tree diagram in Figure 34 below.

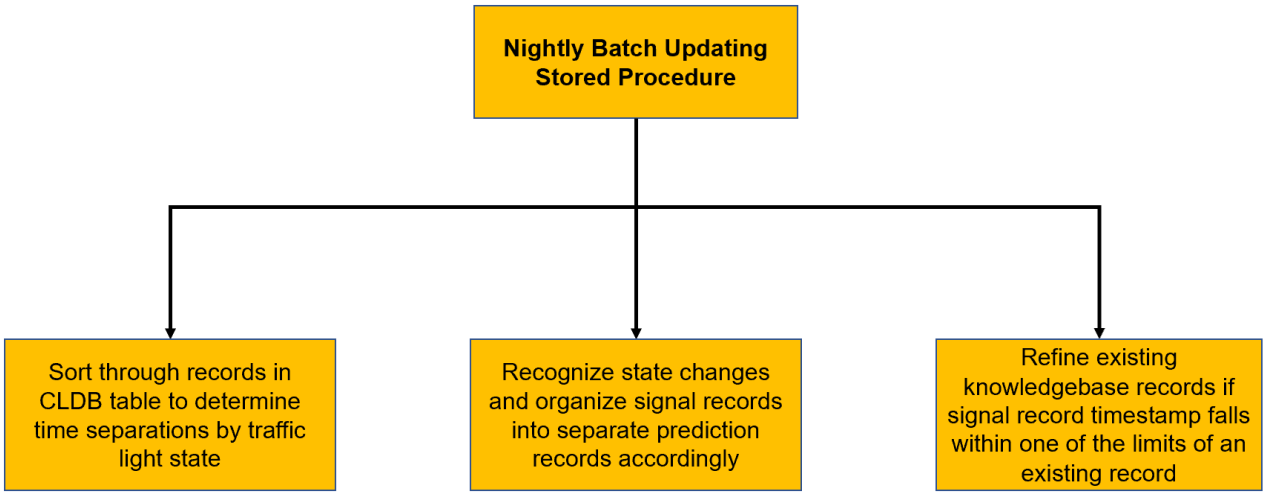

Figure 34 - Function Tree Diagram (Nightly Batch Updating Stored Procedure)

The first main function is to sort through the records in the CLDB to determine their time separations by traffic light state. This is done by setting a time frame in which traffic lights will not have changed state twice; i.e. for example, change state from green to red back to green within a certain time period. This time frame is assumed to be 10 seconds to be conservative and additionally more accurate in determining signal length predictions. To illustrate the first functionality of the batch updates procedure, an example set of signal records and their corresponding KB records are represented below in Figure 35.

Signal Records		
300418	14:34:58	Green
300418	14:35:01	Green
300418	14:35:22	Green

Knowledgebase Records			
300418	14:34:58	14:35:01	Green
300418	14:35:22	14:35:22	Green

Figure 35 - Example Signal Records and Corresponding KB Records (First Function of Batch Updating Procedure)

It can be seen that for the first two signal records above, the green light state was observed twice within three seconds of one another, allowing both records to be consolidated as

one signal length prediction record in the KB, due to being within the time threshold of 10 seconds. However, the third record was uploaded more than 10 seconds after the second record was uploaded, indicating that the signal state, although still observed to be green, could have changed twice from green to red back to green, within the time frame. This requires a new KB record to be inserted to correspond to this signal record to maintain high prediction accuracy.

The second function of the batch updates procedure is to recognize state changes when developing a signal length prediction. This is performed by again inserting a new KB record every time a state change is observed, to prevent the signal length prediction from combining the durations of a green and red traffic light state. An example set of signal records and their corresponding KB records are shown in Figure 36 to better understand this functionality.

Signal Records		
300418	14:35:26	Green
300418	14:35:28	Green
300418	14:35:29	Red

Knowledgebase Records			
300418	14:35:26	14:35:28	Green
300418	14:35:29	14:35:29	Red

Figure 36 - Example Signal Records and Corresponding KB Records (Second Function of Batch Updating Procedure)

Here, the first two signal records observe the same traffic light state (green) and are within two seconds of one another, so their durations can be encompassed together into one KB prediction record. The third signal record observes the red state, and hence, although it was uploaded within one second of the previous record, it's duration cannot be added to the first KB prediction record. It must be listed as a separate KB record, as can be seen in Figure 36.

The last main function of the procedure is to recognize KB records that already exist at a particular time frame; to refine instead of entirely recreate signal length predictions. This is done

by checking whether the new signal record's timestamp falls within the time limits of any existing KB records. If it does, then the existing KB record time limits do not have to be modified. If it doesn't, the procedure moves on to check whether the timestamp is within 10 seconds (time threshold assumption stated earlier in this section) before or after any of the KB records' time period limits. If it satisfies this condition, the record is refined by extending its upper or lower limit, based on where the timestamp falls with respect to the original limits. An example of this is provided in Figure 37 for clarity.

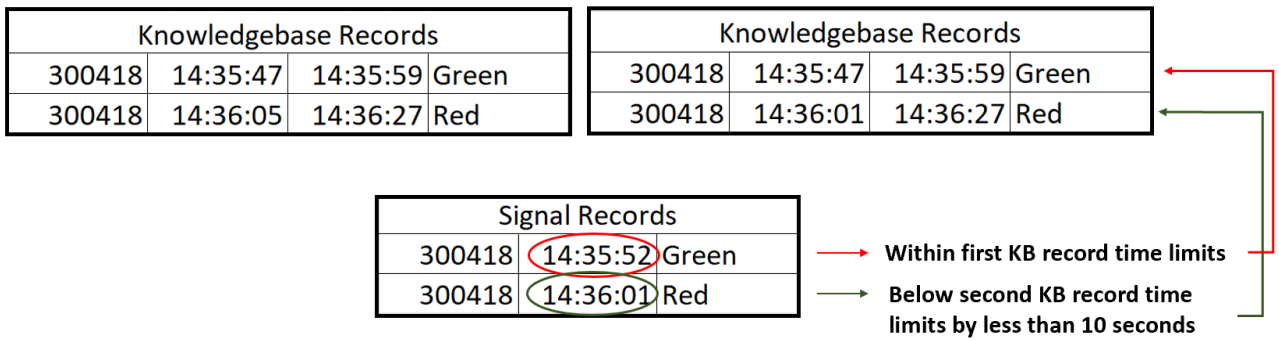

Figure 37 - Example Signal Records and Corresponding KB Records (Third Function of Batch Updating Procedure)

The green state is observed at 14:35:52, which is right between the existing KB record limits for that particular signal ID (14:35:47 – 14:35:59). This means that the signal record's duration is already observed in the existing prediction, hence preventing the need to make any modifications to the KB record. On the other hand, the red state is observed at 14:36:01, which is four seconds away from the lower limit of the existing KB record. Since this is within the time threshold of 10 seconds, guaranteeing that the traffic light state would not have changed twice within this period, the duration of the signal record can be added to the lower limit of the existing prediction, and the KB record can be refined to offer a more inclusive prediction to the driver, as shown in Figure 37.

The batch updates procedure allows signal lengths to be calculated and stored in a knowledgebase where they can be easily accessed real-time during a drive. The next subdivision will discuss how signal lengths are relayed back to the driver on the GUI.

In order to relay signal length predictions back to a driver real-time, the KB is contacted when the driver is within a particular threshold distance of the traffic light. This is done by executing a query that searches for KB records corresponding to only the particular signal ID shown as closest to the driver's current location.

The query retrieves the lower time limit, upper time limit, and expected traffic light state columns from the KB, for all records that correspond to a particular signal ID. The lower and upper time limit values are converted by their hour, minute, and second parameters into datetime.time objects that can be used to make time comparisons in Python. The current system time, which represents the current time at which the driver is approaching the traffic light, is found using the datetime.now object. This is compared with the converted lower and upper limit values of each KB record to determine if a known record exists at the current time frame. If it does, the remaining time until the signal changes state is calculated by subtracting the current time from the upper time limit. Since only hours, minutes, and seconds are represented in each time stamp, Python can convert the times to a 24 hour clock system using the datetime.time object and find the appropriate difference in time. Once the signal change time has been calculated, the expected traffic light state associated with the record is read into the program and stored in a variable to be used in the GUI. To display the predicted signal length update, another text widget is added to the GUI. This widget will display the following text to alert the driver about the signal change time and expected state:

“Signal State will remain ____ for approximately _____.”

In the event that the current time stamp is not found within the lower and upper limit of any KB record time periods, the GUI will display the following message on the same text widget: “Signal State cannot be predicted with accuracy for this traffic light.”

This is to alert the driver that a prediction cannot be offered with confidence on the signal change time or expected state, since there is not enough learning data to support it. As the car passes through the traffic light, the state it observes along with the current time stamp will be uploaded into the CLDB and used to create a new KB record for this particular time period. This will allow future vehicles passing through the same traffic light at this time period to receive signal length predictions. Figure 38 shows a flowchart for this process.

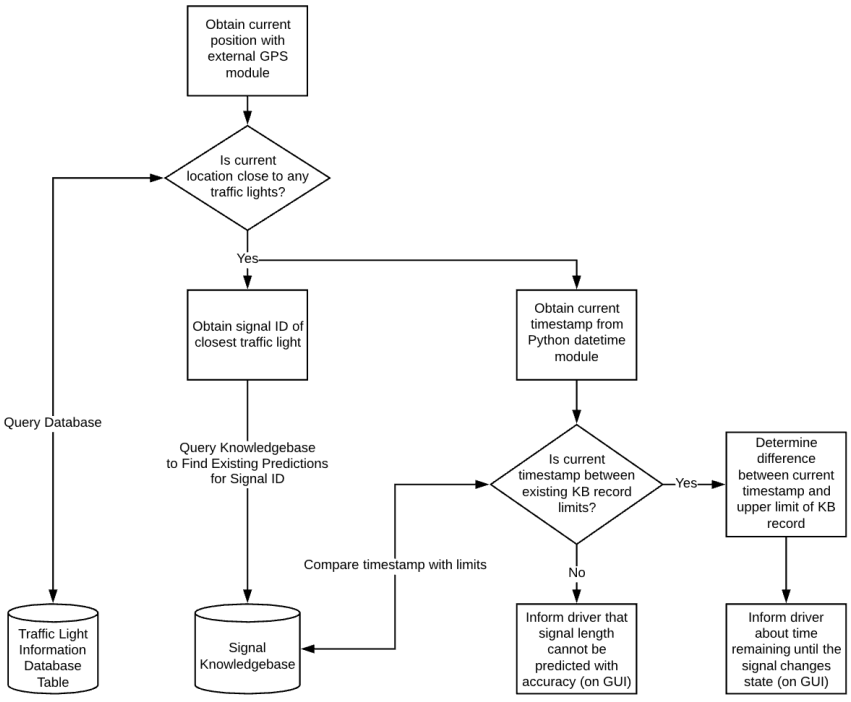

Figure 38 - Determining and Relaying Signal Length Predictions Back to a Driver (Flowchart)

The batch procedure allows for self-learning and adjustment of KB record time periods to reflect learning data acquired daily, making its accuracy in offering predictions extremely high.

The next section of this chapter will present the design of the entire integrated system and focus on the method developed to integrate the system.

Section 3.5: Integrated System

3.5.1: Design of the Integrated System

The integrated system is represented in the flowchart shown at the beginning of this chapter (Figure 15). It was developed by combining all of the sub-modules discussed in sections 3.3 and 3.4. The system allows for the two main algorithms, traffic light learning and prediction, to be executed in parallel although they don't technically rely on each other, since both depend on the position of the vehicle as identified by the GPS Receiver. The learning algorithm uses the GPS location to identify the traffic light detected by the model, in order to authorize signal record uploads to the correct signal ID. In the case of the prediction algorithm, the GUI updates to the driver are offered real-time at a specified distance to the traffic light on the driver's route, which is calculated by the GPS Receiver. For these reasons, the system is integrated and deployed in one simplified process that simultaneously executes both algorithms while polling the GPS only once every iteration to acquire vehicle position data.

3.5.2: Methods Developed to Integrate the System

In order to run both the traffic light learning and prediction algorithms in parallel to integrate the system, multiple methods were developed and tested in Python. These include the multiprocessing and threading packages method, the database read/write method, and the continuous loop integration method. The former two methods will be detailed in Section B of the Appendix. The next subsection will provide an overview of the continuous loop integration

method, which was the final chosen method to integrate the traffic light learning and prediction system.

The continuous loop integration method was originally avoided due to the fact that the traffic light learning and prediction systems are simultaneously executed, as explained in section 3.5.1. In this method, both functions, GPS polling and driver updates as well as live-stream video recognition, are under one while loop, making them execute one after another. This allows variable values to be transferred and used between both functions without the need to process them as queues or through a database, as explained in sections B.1 and B.2 of the Appendix.

Although the processes are run one after another, the delay in performing live-stream video recognition while displaying driver updates and polling the GPS is minimal (to a millisecond value), and hence can be ignored for testing purposes. This method worked the best of the three methods in integrating the whole system and hence was used in the final prototype developed and delivered at the end of the project.

The testing process conducted and results obtained using this approach will be delved into and discussed in the next chapter of this document. The final section of this chapter will discuss the driver display unit; a graphical user interface developed to relay updates to the driver real-time.

Section 3.6: Driver Display Unit

Traffic light state detections and signal length predictions are extremely useful information to provide to leader and follower vehicle drivers real-time. Since this project was tested using an HP Envy x360 Convertible Laptop PC with Microsoft Windows 10 Pro as the

predominant operating system, a GUI to deliver the above-mentioned updates to the driver was developed. A visual of the GUI is presented below in Figure 39.

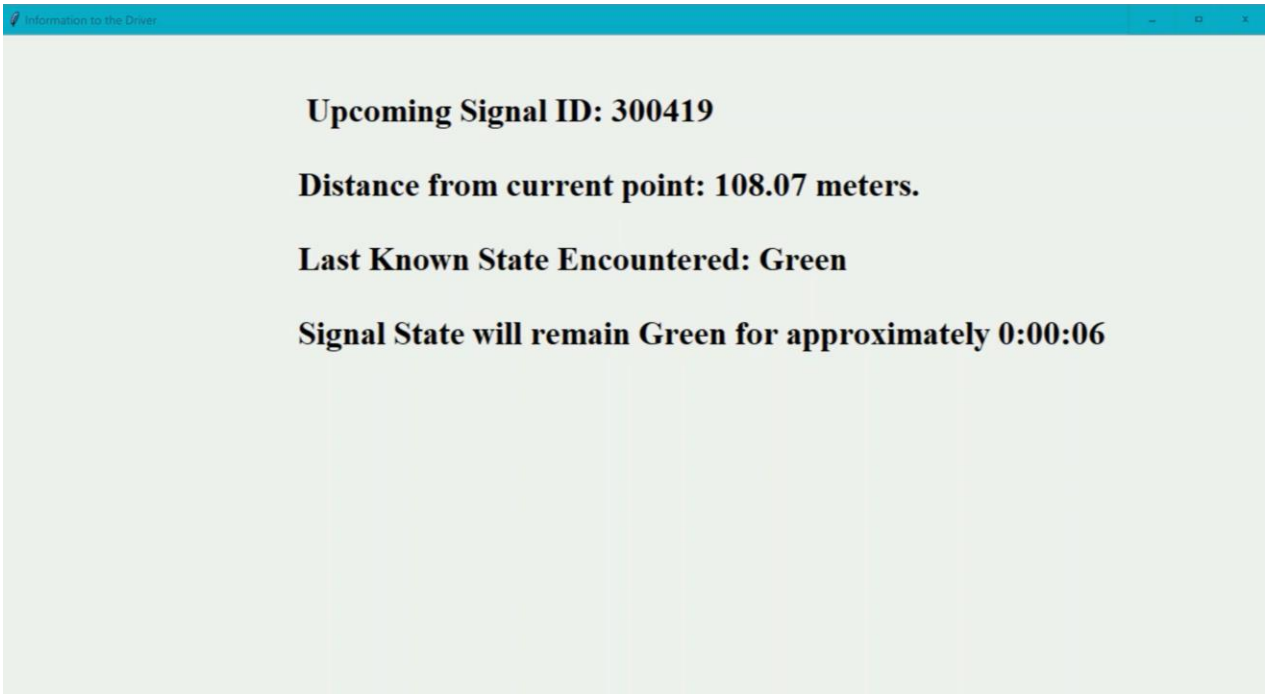

Figure 39 - Driver Display Unit (GUI)

The following sub-section will discuss how this GUI was created.

3.6.1: Graphical User Interface (GUI) to Display Updates Real-Time to a Driver

In order to develop this graphical user interface, the easygui package developed for Python with the timerbox module was utilized. This package allows the creation of a GUI root window designed with customized text widgets that can allow information to be displayed to a user. The root window can be created in specified dimensions as per the user's application. In this case, a root window geometry of 1265 x 700 pixels (full screen) was specified, since the main purpose of the GUI is to show updates to a driver real-time; hence the font size must be clear enough and big for the driver to read. If multiple GUIs are being opened within the same

application to serve different purposes, each GUI's root window can be named with a different title. This is to help the user understand the difference in the information being displayed to them. Only one GUI was developed in this project, and it is titled "Information about the Upcoming Traffic Light on my Route", in order to alert the driver that they are receiving updates about a traffic light they are going to cross. There are multiple text widgets displayed on the GUI providing a variety of useful information to the driver. These include the upcoming traffic light on a driver's route (discussed in section 3.4.2), the distance from the driver's current location to the light (found by the finding the nearest traffic lights module, discussed in section 3.4.1), the last known state of the light (discussed in section 3.4.3), and if available, a prediction on the duration of its current state (discussed in section 3.4.4). All of this information is helpful in assisting a driver to make a decision to either stay on the current route or switch to a less congested route of travel, as well as increase their speed to pass through a green or slow down to approach a red. The GUI is scheduled to appear for three seconds and then disappear, so that it doesn't abort or delay the main program execution. Figure 40 below shows the integrated system at work: the traffic light learning system is detecting light states on live video stream and uploading it as signal records into the central learning database tables (shown by the video capture frame and signal record insert print statements on the Python console); the traffic light prediction system is providing the driver updates on upcoming signal ID, last known state, and signal length predictions on the GUI described in this section.

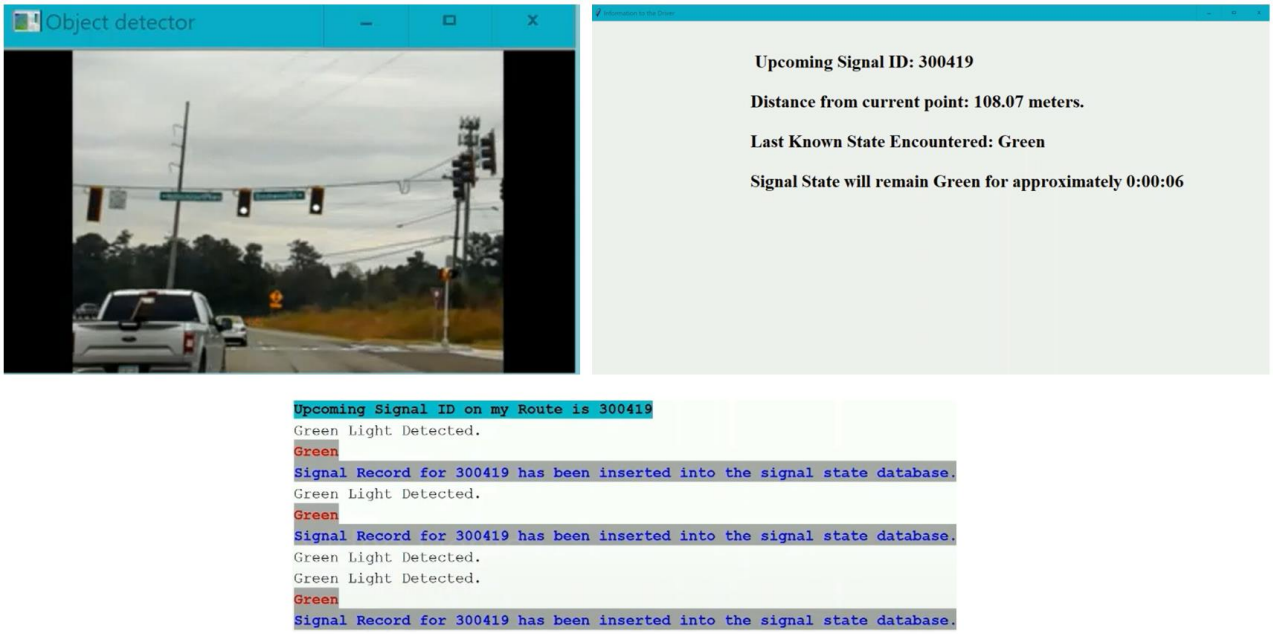

Figure 40 - Integrated System Executed During a Live Test Run

This concludes the methods section of the document. The next chapter will discuss the testing processes employed to determine the success of the integrated system and the corresponding results.

CHAPTER 4. RESULTS & DISCUSSION

This chapter starts off with a summary of the testing conditions provided to determine the relative success of the final integrated system discussed in Section 3.6 of the document. It then proceeds to display the driving routes used for all testing runs performed to validate the system. Finally, the results from live road testing of traffic light recognition model 2, traffic light recognition model 3, and the integrated system are presented.

4.1: Summary of the Required Hardware, Software, and Testing Personnel

The entire system script can be run using a Python IDE, such as IDLE or PyCharm, installed on a laptop. The external hardware required to run the system are the GlobalSat BU-353 GPS Receiver and a Logitech Brio 4K UHD Webcam. The GPS Receiver is connected to the USB Port specified as the COM port within the main script. The Webcam is connected to another USB Port on the laptop and positioned using an iPhone suction cup holder attached to the inner side of the windshield. The positioning of the webcam matters the most as it determines the angle at which the video frame is captured and submitted to the model for detection. A properly captured video frame allows for more accurate traffic light state detections and better overall performance of the integrated system. Figure 41 represents the in-vehicle set-up to perform live road testing.

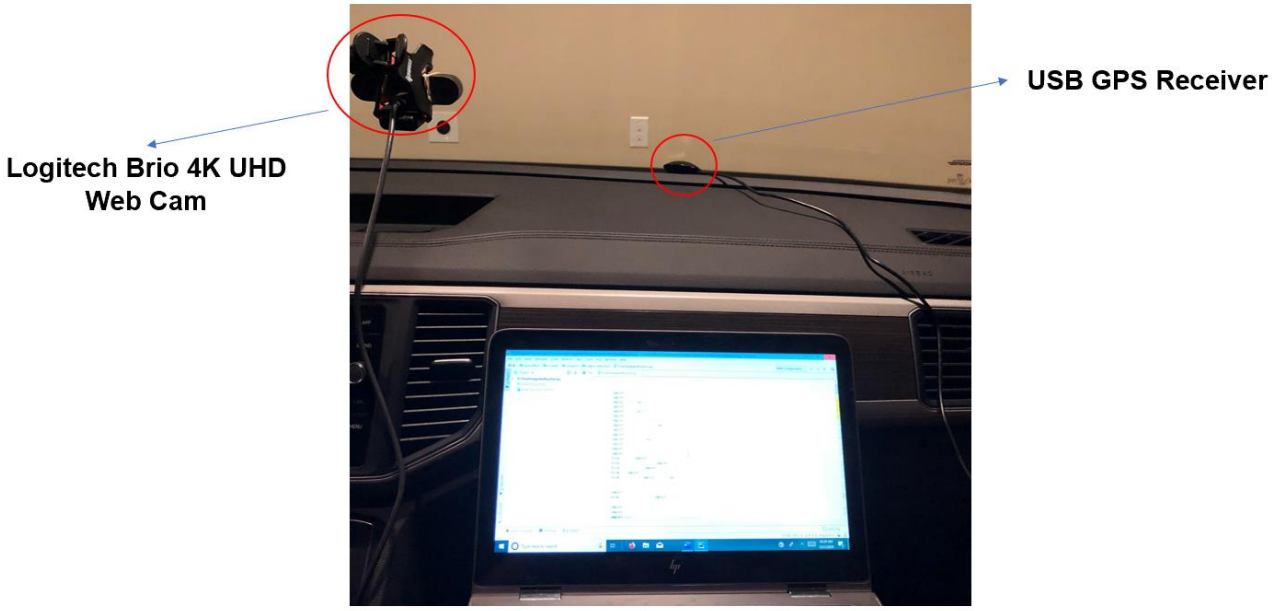

Figure 41 - In-Vehicle Setup to Perform Live Road Testing

Two people were required to perform the testing process. One person serves as the driver of the vehicle while the other person executes the script in Python, monitors the results obtained from the traffic light recognition model and prediction system, and ensures that the driver updates are properly displayed on the GUI.

The following section will discuss and display the driver routes used to test the traffic light recognition models and integrated system respectively.

4.2: The Driving Routes

A total of three driving routes were used to perform all of the testing runs. The first route was used to primarily test the performance of traffic light recognition model 2. The second route was used to perform all of the remaining test runs; for traffic light recognition model 3 and the integrated system. The third route was additionally used to test both traffic light recognition model performances in Atlanta, GA. Figures 42, 43, and 44 display the first, second, and third

driving routes respectively, with the traffic lights pinpointed as red circles on the routes. Some points on the routes have two circles next to each other, indicating that traffic lights on both sides of the intersection (forward and backward) were used for testing. The first route consisted of 11 traffic lights in one area of Cumming, the second route consisted of 10 traffic lights in another area of Cumming, and the third route consisted of 12 traffic lights in midtown Atlanta near the Georgia Tech campus.

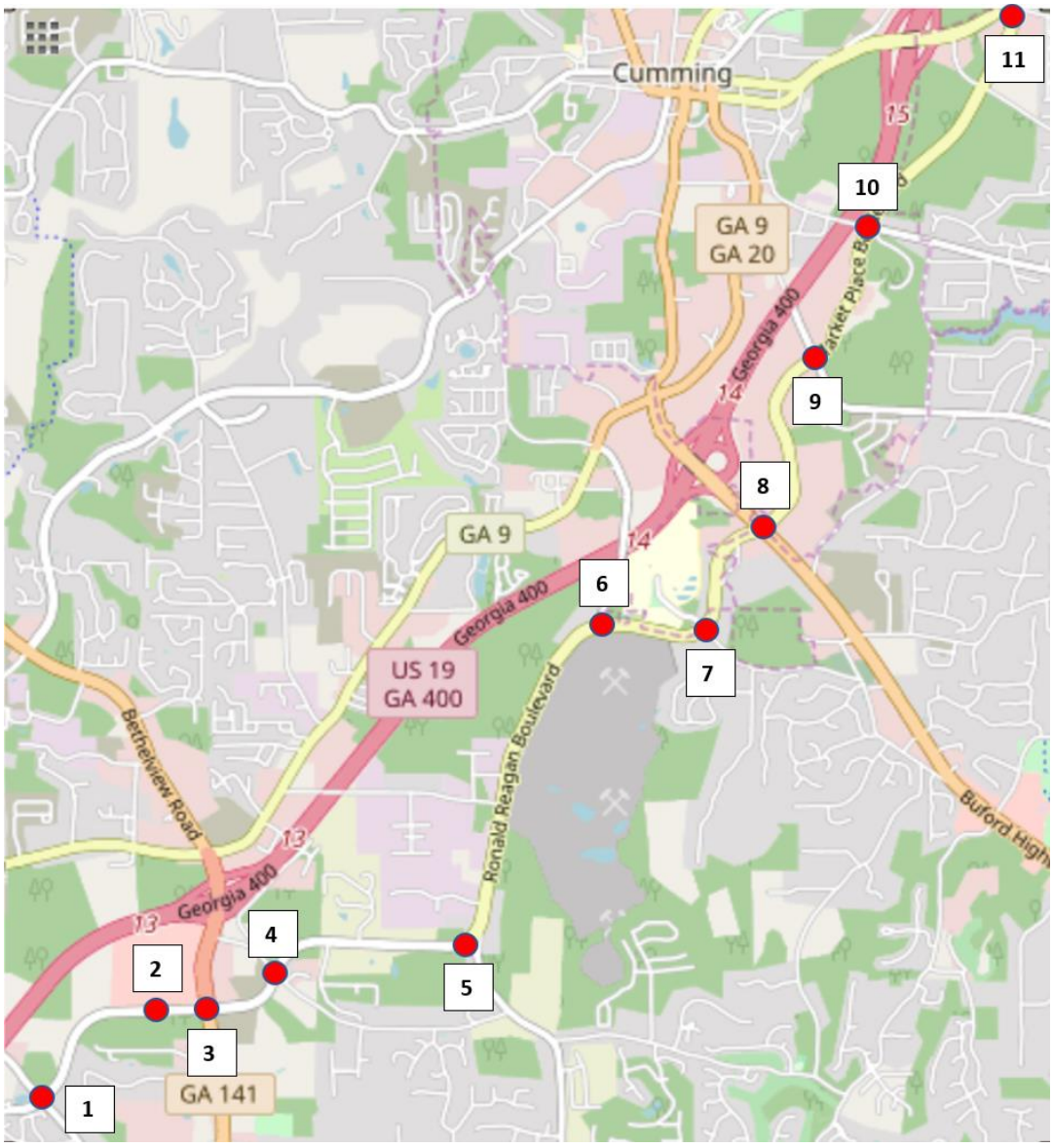

Figure 42 - Driving Route 1 (Cumming, GA)

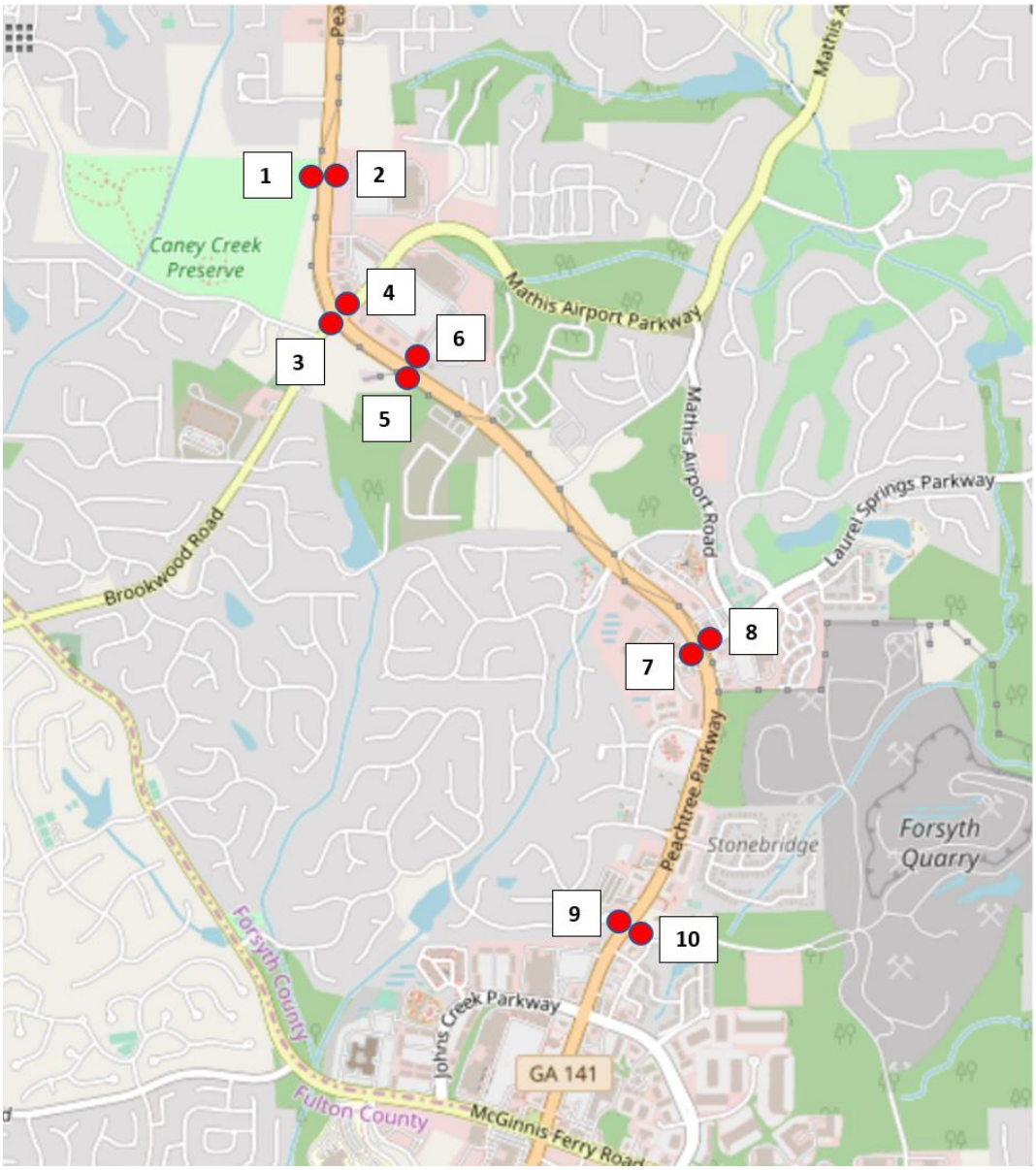

Figure 43 - Driving Route 2 (Cumming, GA)

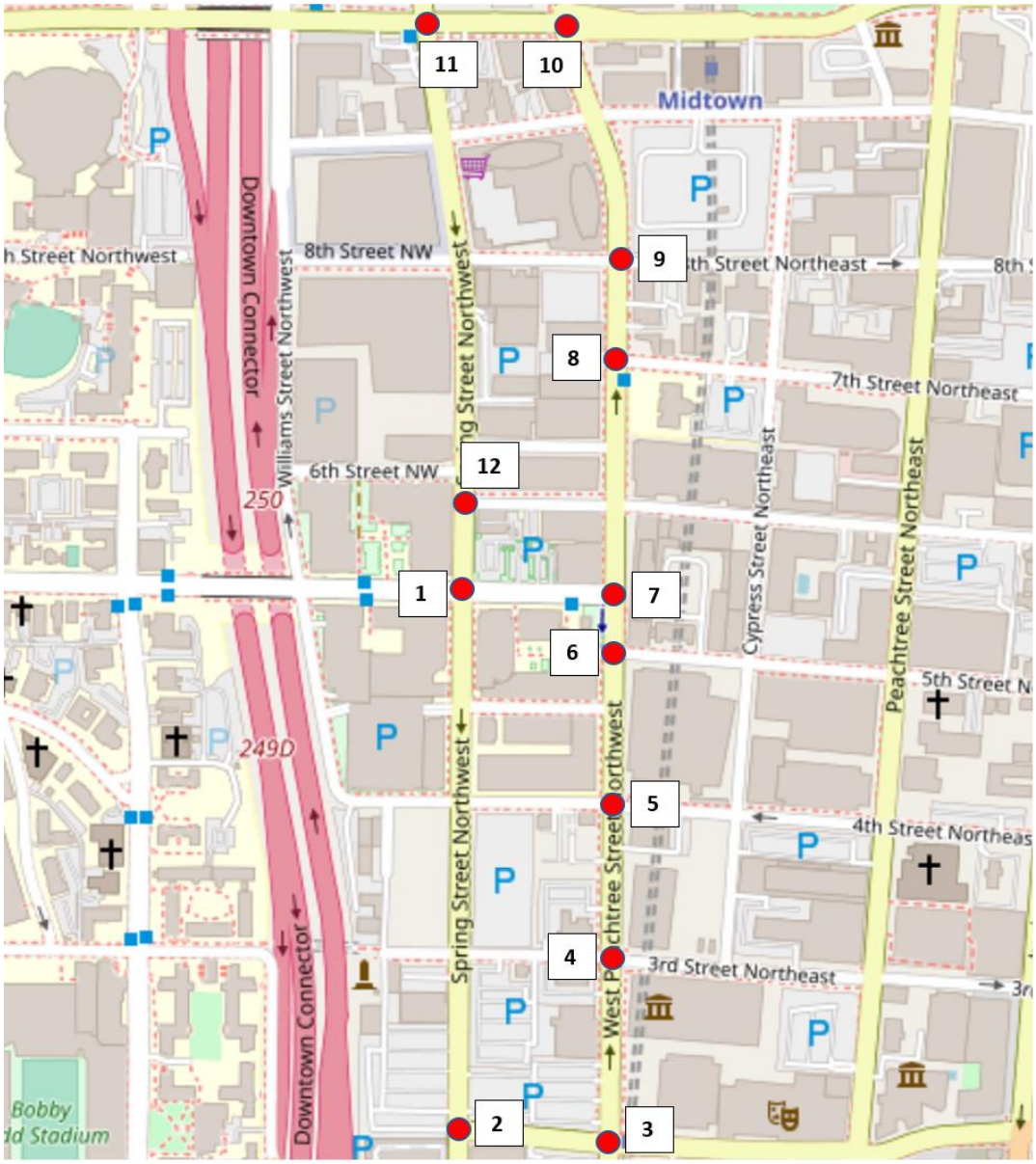

Figure 44 - Driving Route 3 (Atlanta, GA)

The next section will discuss the different configurations of traffic lights found on the above-mentioned driving routes. This will be helpful in understanding the results from testing both traffic light recognition models live on the three routes.

4.3: The Different Configurations of Traffic Lights Found on Driving Routes 1 – 3

Each driving route used for testing had traffic lights with various configurations. Driving route 1 had a total of two different configurations. These are named as TL 1 and TL 2 for ease of understanding, but they actually represent the Green Light, Green Light 2, Red Light, and Red Light 2 configurations. Driving route 2 had a total of three different configurations, including TL 1 and TL 2. The third configuration, which is named as TL 3 for better understanding, is actually composed of the Green Light 3 and Red Light 3 configurations (according to Model 3 framework; in Model 2, the TL 3 configuration is classified under the TL 1 type). Driving route 3 had the same configurations as driving route 2; TL 1, TL 2, and TL 3. Understanding the differing characteristics of each traffic light configuration is critical in determining model success, the results of which will be presented in the following sections for each model. The next section will focus on the results from testing traffic light recognition model 2 live on driving route 1.

4.4: Traffic Light Recognition Model 2 Live Test Run Results (Driving Route 1)

Traffic Light Recognition Model 2 was tested on frozen frames of live stream video capture (image detection). The list of traffic lights represented by their signal IDs, their configurations and respective states as encountered during the test run (in Cumming), whether the model detected the states correctly, the model's confidence in detection, and the number of mis-detections corresponding to yellow lights are all represented in Table 2. The test was administered on May 26, 2019 at 2:30 p.m. The weather was sunny so the lighting conditions were extremely bright. The results of this test run will be analyzed in section 4.9 which discusses the final model selection. The following section will delve into the results from testing traffic light recognition model 2 live on driving route 3.

Table 2 - Model 2 Live Test Run Data (Driving Route 1 – Cumming, GA)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Did Model Detect State Correctly?	Model Confidence in State Detection	Number of Mis-Detections Corresponding to Yellow Lights
1	(TL 2/TL 1) - Red (TL 2/TL 1) - Green	Yes Yes	95% 92%	0 0
2	(TL 2/TL 1) - Green	Yes	97%	0
3	(TL 1/TL 1) - Red (TL 1/TL 1) - Green	No Yes	23% 73%	1
4	(TL 2/TL 1) - Green	Yes	91%	0
5	(TL 2/TL 1) – Red (TL 2/TL 1) - Green	Yes Yes	94% 92%	0 0
6	(TL 2/TL 1) – Green	Yes	89%	0
7	(TL 1/TL 1) – Red (TL 1/TL 1) – Green	No Yes	19% 65%	1 0
8	(TL 2/TL 1) - Green	Yes	98%	0
9	(TL 2/TL 1) – Green	Yes	93%	0
10	(TL 1/TL 1) – Red (TL 1/TL 1) - Green	Yes Yes	88% 91%	0 0
11	(TL 2/TL 1) – Green	Yes	96%	0

4.5: Traffic Light Recognition Model 2 Live Test Run Results (Driving Route 3)

Table 3 lists the traffic lights represented by their signal IDs, their configurations and respective states as encountered during the test run (in Atlanta), whether the model detected the states correctly, the model's confidence in detection, and the number of mis-detections corresponding to yellow lights. The test was administered on May 28, 2019 at 2:30 p.m. The weather was cloudy and foggy since it had rained a couple of hours before the test, so the lighting conditions were relatively dull. The results of this test run will be analyzed in section

4.9. The next section will discuss the results from testing traffic light recognition model 3 live on driving route 1.

Table 3 - Model 2 Live Test Run Data (Driving Route 3 – Atlanta, GA)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Did Model Detect State Correctly?	Model Confidence in State Detection	Number of Mis-Detections Corresponding to Yellow Lights
1	(TL 2/TL 1) - Green	Yes	76%	0
2	(TL 2/TL 1) - Green	Yes	85%	0
3	(TL 3/TL 3) - Red (TL 3/TL 3) - Green	No No	16% 24%	0 0
4	(TL 2/TL 1) - Green	Yes	54%	0
5	(TL 2/TL 1) – Red (TL 2/TL 1) - Green	No No	8% 23%	1 0
6	(TL 2/TL 1) – Red (TL 2/TL 1) - Green	No Yes	11% 61%	1 0
7	(TL 3/TL 3) – Red (TL 3/TL 3) – Green	No No	27% 17%	0 0
8	(TL 2/TL 1) - Green	No	18%	0
9	(TL 2/TL 1) – Green	No	2%	0
10	(TL 2/TL 1) – Red (TL 2/TL 1) - Green	No No	7% 31%	1 0
11	(TL 3/TL 3) – Green	No	15%	0
12	(TL 2/TL 1) – Green	Yes	72%	0

4.6: Traffic Light Recognition Model 3 Live Test Run Results (Driving Route 1)

Traffic Light Recognition Model 3 was tested similarly to Model 2; live stream video capture frames were frozen and the model was run on them. The list of traffic lights represented by their signal IDs, their configurations and respective states as encountered during the test run

(in Cumming), whether the model detected the states correctly, the model's confidence in detection, and the number of mis-detections corresponding to yellow lights are all represented in Table 4, shown on the next page. The test was administered on October 7, 2019 at 2:30 p.m. The weather was relatively sunny so the lighting conditions were bright. The results of this test run will be analyzed in section 4.9 which discusses the final model selection. The next section will focus on the results from testing traffic light recognition model 3 live on driving route 2.

Table 4 - Model 3 Live Test Run Data (Driving Route 1 – Cumming, GA)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Did Model Detect State Correctly?	Model Confidence in State Detection	Number of Mis-Detections Corresponding to Yellow Lights
1	**(TL 2/TL 1) - Green**	**Yes**	**97%**	**0**
2	**(TL 2/TL 1) - Green**	**Yes**	**96%**	**0**
3	**(TL 1/TL 1) - Red** **(TL 1/TL 1) - Green**	**Yes** **Yes**	**91%** **99%**	**0** **0**
4	**(TL 2/TL 1) - Red** **(TL 2/TL 1) - Green**	**Yes** **Yes**	**91%** **97%**	**0** **0**
5	**(TL 2/TL 1) - Green**	**Yes**	**94%**	**0**
6	**(TL 2/TL 1) – Green**	**Yes**	**98%**	**0**
7	**(TL 1/TL 1) – Red** **(TL 1/TL 1) – Green**	**Yes** **Yes**	**95%** **93%**	**0** **0**
8	**(TL 2/TL 1) - Green**	**Yes**	**97%**	**0**
9	**(TL 2/TL 1) – Green**	**Yes**	**97%**	**0**
10	**(TL 1/TL 1) - Green**	**Yes**	**94%**	**0**
11	**(TL 2/TL 1) – Green**	**Yes**	**92%**	**0**

4.7: Traffic Light Recognition Model 3 Live Test Run Results (Driving Route 2)

Table 5 lists the traffic lights represented by their signal IDs, their configurations and respective states as encountered during the test run (in Cumming), whether the model detected the states correctly, the model's confidence in detection, and the number of mis-detections corresponding to yellow lights. The test was administered on October 7, 2019 at 2:30 p.m. The weather was relatively cloudy so the lighting conditions were dull. The results of this test run will be analyzed in section 4.9. The following section will delve into the results from testing traffic light recognition model 3 live on driving route 3.

Table 5 - Model 3 Live Test Run Data (Driving Route 2 – Cumming, GA)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Did Model Detect State Correctly?	Model Confidence in State Detection	Number of Mis-Detections Corresponding to Yellow Lights
1	**(TL 1/TL 1) - Green**	**Yes**	**96%**	**0**
2	**(TL 3/TL 3) – Red** **(TL 3/TL 3) - Green**	**Yes** **Yes**	**81%** **94%**	**0** **0**
3	**(TL 3/TL 3) - Green**	**Yes**	**92%**	**0**
4	**(TL 1/TL 1) – Red** **(TL 1/TL 1) - Green**	**Yes** **Yes**	**97%** **99%**	**0** **0**
5	**(TL 2/TL 1) – Red** **(TL 2/TL 1) - Green**	**Yes** **Yes**	**93%** **95%**	**0** **0**
6	**(TL 2/TL 1) – Green**	**Yes**	**90%**	**0**
7	**(TL 1/TL 1) – Red** **(TL 1/TL 1) – Green**	**Yes** **Yes**	**96%** **96%**	**0** **0**
8	**(TL 3/TL 3) - Green**	**Yes**	**91%**	**0**
9	**(TL 3/TL 3) – Green**	**Yes**	**88%**	**0**
10	**(TL 3/TL 3) - Green**	**Yes**	**87%**	**0**

4.8: Traffic Light Recognition Model 3 Live Test Run Results (Driving Route 3)

The list of traffic lights represented by their signal IDs, configurations, respective states, whether the model detected the states correctly, detection confidence, and the number of mis-detections corresponding to yellow lights are represented in Table 6. The test was administered on October 8, 2019 at 2:30 p.m. The weather was sunny so the lighting conditions were bright. The results of this test will be analyzed in the next section.

Table 6 - Model 3 Live Test Run Data (Driving Route 3 – Atlanta, GA)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Did Model Detect State Correctly?	Model Confidence in State Detection	Number of Mis-Detections Corresponding to Yellow Lights
1	(TL 2/TL 1) - Red (TL 2/TL 1) - Green	Yes Yes	98% 97%	0 0
2	(TL 2/TL 1) - Green	Yes	99%	0
3	(TL 3/TL 3) - Green	Yes	92%	0
4	(TL 2/TL 1) - Green	Yes	96%	0
5	(TL 2/TL 1) – Red (TL 2/TL 1) - Green	Yes Yes	95% 98%	0 0
6	(TL 2/TL 1) - Green	Yes	96%	0
7	(TL 3/TL 3) – Green	Yes	89%	0
8	(TL 2/TL 1) - Red (TL 2/TL 1) - Green	Yes Yes	94% 98%	0 0
9	(TL 2/TL 1) - Red (TL 2/TL 1) – Green	Yes Yes	94% 97%	0 0
10	(TL 2/TL 1) - Green	Yes	93%	0
11	(TL 3/TL 3) – Green	Yes	84%	0
12	(TL 2/TL 1) – Green	Yes	99%	0

4.9: Discussion: Traffic Light Recognition Model Final Selection

Comparing the performance of both traffic light recognition models, tested at the same time of day and in similar weather conditions, it is clear that Model 3 is a better model to use for the integrated system than Model 2. As can be seen from the state detection success columns in Tables 4 - 6, Model 3 did not miss any detections nor mis detect any traffic light. It also had an average confidence of over 90% in detecting each traffic light state when tested in Cumming and in Atlanta. This is not the case with Model 2, which mistook two yellow light phases (for signals 3 and 7 respectively) as red lights when tested on Driving Route 1, and further exhibited extremely poor performance on Driving Route 3. Additionally, Model 2 also had a lower average confidence level in detecting each traffic light state when tested in Cumming and in Atlanta, compared to Model 3.

One reason for Model 2's lower performance in Cumming in comparison to Model 3's performance is due to the amount of training undergone by the model. Model 2 was trained on approximately 1200 images and underwent rigorous training for 7 weeks, while Model 3 was trained on more than double the number of images (~2500 images), and underwent 5 additional weeks of training. The neural network refines model detection capability as more images are added to its training set, which is why Model 3 had better confidence and lower misses in detection compared to Model 2. This is also why Model 3 did not confuse red lights and yellow lights when performing detection on live stream photo capture unlike Model 2; it was able to better understand and distinguish amongst the colors to detect only red light configurations that it had been trained to detect and ignore yellow lights.

While this theory applies for driving route 1 in Cumming as well as the mistaken yellow light detections on driving route 3 in Atlanta, other reasons for why Model 2's performance was

below average in Atlanta were the configurations it was trained to detect and the environmental surroundings amongst which it was trained to identify a traffic light. The traffic light configurations column in Table 3 proves this fact by showing that signals 3, 7, and 11 on driving route 3 represented the third traffic light configuration. Referring back to the Appendix, it can be seen that Model 2 was not trained to identify this traffic light configuration. Hence, it is possible that at least for those particular traffic lights, Model 2 could have been confused due to identifying a similar shape to its already trained traffic light configurations but not an exact match to any of them. This would have led to the lower confidence and completely missed detections of those lights. Since these configurations of traffic lights are more prevalent in larger cities, it was decided to add these to the Model 3 framework, and by doing so, detections of these lights were largely improved as can be seen through the data in Tables 5 and 6.

The other factor that took more prevalence in Atlanta over Cumming is the background surroundings of the traffic lights. Figure 45 represents the performances of Model 2 and Model 3 on a traffic light image captured in Atlanta, to highlight the better detection capability of Model 3.

Figure 45 - Model 2 and Model 3 Performances on a Traffic Light Image Captured in Atlanta

Midtown Atlanta is filled with tall buildings, trees, and a lot of other background distractions. Cumming, on the other hand, is a small town in the northern Georgia area, with clear backgrounds, and not filled with tall buildings or trees. Since Model 2 was trained primarily on traffic light images in Cumming, it had a better understanding of identifying traffic lights among clear sky backgrounds with no other distractions. In fact, the model was built in such a way that it took the background as a factor in identifying a traffic light. When testing in Atlanta, the model didn't do a good job since it was only able to distinguish a few traffic lights amongst clear sky backgrounds with configurations it had been trained to recognize. The other traffic lights which it had been trained to understand were hidden by skyscraper building backgrounds and trees, making it hard for the model to identify them as an actual traffic light, let alone narrowing the detection down to a particular configuration and traffic light state. Understanding this phenomenon helped improve the framework for Model 3, by training with more Atlanta traffic light images to incorporate the effect of busy backgrounds. Overall, when testing Model 3 in Atlanta (driving route 3), it showed a much better detection capability compared to Model 2, especially on already well-trained traffic light configurations, since it understood how to distinguish the traffic light from its surrounding environment.

4.10: Integrated System Live Test Run Results (Day 1)

The integrated system combines both the traffic light learning and prediction systems together along with the driver updates offered on the GUI. It performs traffic light learning on live stream video, thereby causing a varying number of detections to be processed for each traffic light state, depending on model capability in detection. This is different from the manner in which the traffic light recognition models were originally tested, since the frames are not frozen

and submitted to the model as images, resulting in only one detection for each image. Obtaining learning data from continuous live stream video capture was intentionally introduced in the integrated system to allow for more accuracy in signal length predictions due to more signal records that can be uploaded for each traffic light.

All of the integrated system live test runs were performed on driving route 2 (Cumming, GA). In this section, the overall performance of the integrated system will be presented as just the traffic light recognition model's performance. This is because no prior traffic light state data exists in the CLDB to provide last known state updates or generate and relay signal length predictions to the driver on the first day of testing. The results of this test run will be discussed in section 4.11.

4.10.1: Traffic Light Recognition Model Performance in the Integrated System (Day 1)

Due to its better performance overall (discussed in section 4.9), Model 3 was used as the final model in the integrated system. Table 7 lists the traffic lights by their signal IDs, their configurations, respective states as encountered during the test run (in Cumming), the model's accuracy in detection, and the number of mis-detections corresponding to yellow lights. The model confidence levels were not shown as bounding boxes on the traffic light in order to avoid driver distraction during the test. The test was administered on November 30, 2019 at 2:30 p.m. The weather was cloudy and dark due to rainfall a few hours prior to the test.

Table 7 - Integrated System Live Test Run Results (Day 1 Model Performance)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Model Accuracy in State Detection (%)	Number of Mis-Detections Corresponding to Yellow Lights
1	(TL 1/TL 1) - Green	12/13 = 92.31%	0
2	(TL 3/TL 3) - Green	14/15 = 93.33%	0
3	(TL 3/TL 3) - Green	11/11 = 100%	0
4	(TL 1/TL 1) - Green	9/9 = 100%	0
5	(TL 2/TL 1) - Green	7/8 = 87.5%	0
6	(TL 2/TL 1) – Green	8/8 = 100%	0
7	(TL 1/TL 1) – Green	18/18 = 100%	0
8	(TL 3/TL 3) – Red (TL3/TL3) - Green	17/18 = 97.62% 24/24 = 100%	0 0
9	(TL 3/TL 3) – Green	10/10 = 100%	0
10	(TL 3/TL 3) - Green	10/10 = 100%	0

4.11: Discussion: Integrated System Live Test Run (Day 1)

From the data presented in Table 7 above, it is clear that the model missed or mis identified at most one detection for each traffic light on the route. This can be primarily attributed to the weather, as the test occurred on a day that was really dark with heavy rains. The overall lighting could have caused it to be really hard to identify the traffic light, as the model was trained to identify the bright green or red color amongst the box of the light, which appeared really dim during this test. This caused some of the green lights to be detected as red lights and vice versa, for signals 1, 2, 5, and 8. On the other hand, although the third traffic light configuration was a new addition to the final model framework and hence not as intensely

trained as the other configurations, the model still had equally good confidence in identifying lights in that configuration as the well-trained traffic light configurations (from Model 2). This can be seen through the data in Table 7. All of the green lights identified in the third traffic light configuration were detected with 100% accuracy, except for one mis detection in signal 2. The red light phase identified in signal 8 also faced only one mis detection, causing its accuracy in state detection to drop from 100 to 97.62%. Additionally, none of the mis-detections performed by the model corresponded to yellow light phases, indicating the model's confidence in distinguishing traffic light states by color. Based off of this data, it can be concluded that with further training of the model to account for different lighting conditions and more of traffic lights under the third configuration, performance can be increased to provide better overall accuracy in state detection.

4.12: Integrated System Live Test Run Results (Day 2)

In this section, the overall performance of the integrated system will be again be presented as the traffic light recognition model's performance. The results of this test run will be discussed in section 4.13.

4.12.1: Traffic Light Recognition Model Performance in the Integrated System (Day 2)

The list of traffic lights represented by their signal IDs, their configurations and respective states as encountered during the test run (in Cumming), whether the model identified the states correctly every time it found a detection, as a percentage of the total number of detections found, and the number of mis-detections corresponding to yellow lights are all represented in Table 8. Again, the model confidence levels were not shown as bounding boxes on the traffic light in order to avoid driver distraction during the test. The test was administered

on December 1, 2019 at 2:30 p.m. The weather was cloudy and dark and there were mild rain showers during the test. The results of this test run will be analyzed in the next section.

Table 8 - Integrated System Live Test Run Results (Day 2 Model Performance)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Model Accuracy in State Detection (%)	Number of Mis-Detections Corresponding to Yellow Lights
1	**(TL 1/TL 1) - Green**	**15/16 = 93.75%**	**0**
2	**(TL 3/TL 3) – Red** **(TL 3/TL 3) - Green**	**78/79 = 98.73%** **23/23 = 100%**	**0** **0**
3	**(TL 3/TL 3) - Red** **(TL 3/TL 3) - Green**	**12/12 = 100%** **25/25 = 100%**	**0** **0**
4	**(TL 1/TL 1) - Green**	**8/8 = 100%**	**0**
5	**(TL 2/TL 1) - Green**	**11/11 = 100%**	**0**
6	**(TL 2/TL 1) – Green**	**11/11 = 100%**	**0**
7	**(TL 1/TL 1) – Red** **(TL 1/TL 1) – Green**	**12/12 = 100%** **33/33 = 100%**	**0** **0**
8	**(TL3/TL3) - Green**	**16/16 = 100%**	**0**
9	**(TL 3/TL 3) – Red** **(TL 3/TL 3) – Green**	**3/3 = 100%** **11/12 = 91.67%**	**0** **0**
10	**(TL 3/TL 3) - Green**	**13/13 = 100%**	**0**

4.13: Discussion: Integrated System Live Test Run (Day 2)

From the data presented in Table 8 above, it can be seen that the performance of the model in day 2 was strikingly similar to its performance in day 1. This can be explained by the similarity in weather conditions across both days. The day 2 test occurred during the rain in order to keep the time at which the test was performed relatively consistent across all days of testing. The overall lighting, therefore, was even more dim than day 1, making the traffic lights hardly

visible. As mentioned during the discussion of day 1 testing results, the model was trained to identify really bright green or red colors amongst the traffic light box. Since visibility was really low and the lights were extremely dim to distinguish, some of the traffic light states were detected incorrectly, as seen in signals 1, 2, and 9 from Table 8 above. When narrowing these down, it can be seen that one green light phase and red light phase under the third traffic light configuration contributed to two of the total mis detections. This can be attributed to the newer addition of these configurations to the final model framework; resulting in not being able to train them as rigorously as the TL 1 and TL 2 configurations to detect amongst all types of weather and lighting conditions. This can be fixed with more training and a larger set of images captured in the TL 3 configuration under different conditions.

Although there were three misdetections in total, the model did a great job overall in accurately identifying traffic light states, with one lesser mis detection and worser weather conditions than day 1 testing. This helps conclude that the model can handle relatively gloomy, rainy, and dark weather conditions reasonably well, without allowing them to impact its detection capability overall. This is a very important observation (backed by data from days 1 and 2 of testing) that can extend the model's usability to bigger cities like Atlanta that experience rainy weather almost every week. Further, none of the mis-detections performed by the model corresponded to yellow lights, just as observed in day 1's testing results, showing that the model had good confidence in distinguishing traffic light states by color.

Another aspect of the integrated system that must be evaluated in this section is the driver updates offered on the GUI. Although there are no quantitative results that can be provided for the GUI performance, it is necessary to determine whether the level of accuracy and timing of

the updates offered to the driver are acceptable. The data from Day 1 testing was uploaded to the CLDB and CLDB copy tables real-time throughout the testing process, allowing for a last known state prediction to be offered to the driver on Day 2. This update was delivered as expected, since it just relies on reading the most recent record from the CLDB copy table for that particular signal ID and delivering it to the driver on the GUI.

A more pressing concern with the driver updates delivered, on the other hand, was associated with the distance at which the driver received information. In Day 2 testing, it was observed that the GUI was almost always invoked at a distance of approximately 50 m, due to the distance threshold and directional constraints set within the program (100 m, driver must be facing the same direction as the traffic light), causing driver updates to be delivered at a time when the driver had almost already crossed the signal. This is of no use to the driver if they want to alter their speed or be alerted of a traffic light on their route.

A potential way to address this concern was determined to be increasing the distance threshold within the program from 100 to 200 m and eliminating the directional constraint. The results of these solutions will be presented and discussed in the next section of this report, focusing on the integrated system's overall performance on Day 3 of testing.

4.14: Integrated System Live Test Run Results (Day 3)

In this section, the overall performance of the integrated system will be presented as two parts: the traffic light recognition model's performance (section 4.14.1) and the signal length predictions offered to the driver (section 4.14.2). The results of this test run will be discussed in the next section.

4.14.1: Traffic Light Recognition Model Performance in the Integrated System (Day 3)

Table 9 lists the traffic lights by their signal IDs, their configurations, respective states as encountered during the test run (in Cumming), model accuracy, and the number of mis-detections corresponding to yellow lights. The test was administered on December 9, 2019 at 2:30 p.m. The weather was cloudy and dark due to rainfall prior to the test. To account for this, the contrast of the webcam was increased through the Logitech BRIO camera settings. The results for this test will be discussed in section 4.15.

Table 9 - Integrated System Live Test Run Results (Day 3 Model Performance)

Signal ID	Traffic Light Configuration(s)/State(s) Encountered	Model Accuracy in State Detection (%)	Number of Mis-Detections Corresponding to Yellow Lights
1	**(TL 1/TL 1) - Green**	**2/2 = 100%**	**0**
2	**(TL 3/TL 3) - Green**	**11/11 = 100%**	**0**
3	**(TL 3/TL 3) - Green**	**10/10 = 100%**	**0**
4	**(TL 1/TL 1) - Green**	**7/7 = 100%**	**0**
5	**(TL 2/TL 1) - Green**	**6/6 = 100%**	**0**
6	**(TL 2/TL 1) – Green**	**9/9 = 100%**	**0**
7	**(TL 1/TL 1) – Red** **(TL 1/TL 1) – Green**	**143/143 = 100%** **21/21 = 100%**	**0** **0**
8	**(TL3/TL3) - Green**	**9/9 = 100%**	**0**
9	**(TL 3/TL 3) – Green**	**7/7 = 100%**	**0**
10	**(TL 3/TL 3) - Green**	**9/9 = 100%**	**0**

4.14.2: Signal Length Predictions Offered to the Driver (Day 3)

The signal length predictions generated using Day 1 testing data (Figures 66 - 67 in section D.1 of the Appendix) is represented in the knowledgebase shown in Figure 46.

signalId	time_from	time_to	expected_state
300418	14:35:49	14:35:55	Red
300418	14:35:56	14:36:02	Green
300419	14:36:24	14:36:47	Green
3004110	14:37:01	14:37:07	Green
3004111	14:38:17	14:38:26	Green
3004112	14:39:15	14:39:27	Green
3004119	14:42:02	14:42:12	Green
3004120	14:43:10	14:43:23	Green
3004121	14:44:58	14:45:14	Green
3004122	14:45:27	14:45:35	Green
3004123	14:45:58	14:46:09	Green

Figure 46 - Knowledgebase (Predictions Formed Using Day 1 Testing Data)

Adding the Day 2 data to this (Figures 68 - 70 in section D.2 of the Appendix) generates the more cumulative knowledgebase shown in Figure 47.

signalId	time_from	time_to	expected_state
300418	14:34:49	14:35:01	Green
300418	14:35:49	14:35:55	Red
300418	14:35:56	14:36:02	Green
300419	14:35:22	14:35:47	Red
300419	14:35:47	14:35:56	Green
300419	14:36:24	14:36:47	Green
3004110	14:36:09	14:36:15	Green
3004110	14:37:01	14:37:07	Green
3004111	14:37:20	14:37:29	Green
3004111	14:38:17	14:38:26	Green
3004112	14:38:15	14:38:25	Green
3004112	14:39:15	14:39:27	Green

signalId	time_from	time_to	expected_state
3004119	14:40:00	14:40:10	Green
3004119	14:42:02	14:42:12	Green
3004120	14:40:59	14:41:09	Green
3004120	14:43:10	14:43:23	Green
3004121	14:42:15	14:42:25	Green
3004121	14:44:58	14:45:14	Green
3004122	14:42:36	14:43:18	Red
3004122	14:43:20	14:43:40	Green
3004122	14:45:27	14:45:35	Green
3004123	14:43:59	14:44:07	Green
3004123	14:44:08	14:44:12	Red
3004123	14:45:58	14:46:09	Green

Figure 47 - Knowledgebase (Predictions Formed Using Day 1 and Day 2 Testing Data)

Figure 48 below shows the times at which the driver first encountered and received updates on the signals during Day 3 of testing.

Signal Encounters
300418 – 14:34:31
300419 – 14:35:50
3004110 – 14:36:06
3004111 – 14:37:22
3004112 – 14:38:19
3004119 – 14:40:35
3004120 – 14:42:05
3004121 – 14:43:51
3004122 – 14:45:31
3004123 – 14:46:06

Figure 48 - Signal Encounter Times (Day 3 Testing)

Table 10 shows the predictions relayed back to the driver on the GUI based on the times shown in Figure 48 and the knowledgebase presented in Figure 47. If a prediction was not available for a particular traffic light at the current time but existed within an hour of it, the signal length at that time is relayed back to the driver. In these cases, it was ensured that the driver was notified that the signal's timings apply to that particular time and not the current time, so that they can make educated decisions about altering their speeds to pass through or wait at the light.

Table 10 - Predictions Relayed Back to Driver Real-Time (Day 3 Testing)

300418	Signal State will be Green for approximately 0:00:12 at 14:34:49
300419	Signal State will remain Green for approximately 0:00:06
3004110	Signal State will be Green for approximately 0:00:06 at 14:36:09
3004111	Signal State will remain Green for approximately 0:00:07
3004112	Signal State will remain Green for approximately 0:00:06
3004119	Signal State will be Green for approximately 0:00:10 at 14:42:02
3004120	Signal State will be Green for approximately 0:00:13 at 14:43:10
3004121	Signal State will be Green for approximately 0:00:16 at 14:44:58
3004122	Signal State will remain Green for approximately 0:00:04
3004123	Signal State will remain Green for approximately 0:00:03

The final testing data from Day 3 is shown in Figures 71 - 72 in section D.3 of the Appendix. It adds in the records highlighted in yellow in Figure 49 to the existing knowledgebase shown in Figure 47 to complete the knowledgebase developed for testing.

signalId	time_from	time_to	expected_state
300418	14:34:31	14:34:38	Green
300418	14:34:49	14:35:01	Green
300418	14:35:49	14:35:55	Red
300418	14:35:56	14:36:02	Green
300419	14:35:22	14:35:47	Red
300419	14:35:47	14:35:56	Green
300419	14:36:24	14:36:47	Green
3004110	14:36:06	14:36:15	Green
3004110	14:37:01	14:37:07	Green
3004111	14:37:20	14:37:30	Green
3004111	14:38:17	14:38:26	Green
3004112	14:38:15	14:38:33	Green
3004112	14:39:15	14:39:27	Green
3004119	14:40:00	14:40:10	Green
3004119	14:40:35	14:40:46	Green

signalId	time_from	time_to	expected_state
3004119	14:42:02	14:42:12	Green
3004120	14:40:59	14:41:09	Green
3004120	14:42:05	14:42:52	Red
3004120	14:42:53	14:42:57	Green
3004120	14:43:10	14:43:23	Green
3004121	14:42:15	14:42:25	Green
3004121	14:43:51	14:43:59	Green
3004121	14:44:58	14:45:14	Green
3004122	14:42:36	14:43:18	Red
3004122	14:43:20	14:43:40	Green
3004122	14:45:27	14:45:42	Green
3004123	14:43:59	14:44:07	Green
3004123	14:44:08	14:44:12	Red
3004123	14:45:58	14:46:13	Green

Figure 49 - Knowledgebase (Predictions Formed Using Day 1, Day 2, and Day 3 Testing Data)

4.15: Discussion: Integrated System Live Test Run (Day 3)

The model performance in day 3 was the best of all three days of testing. There were no mis detections or missed detections, despite the weather conditions being very similar to days 1 and 2 of testing (all testing for the integrated system was done the same week). While this can be seen as a "lucky" day of testing, data from Table 9 can confirm that the model actually performed a lot of detections, even more than on day 1 and day 2 of testing, and still was able to distinguish the state of the traffic light properly each and every time. This shows that increasing the webcam contrast on days with dark and gloomy weather, such as the three days of testing depicted in this document, will allow for better model recognition of traffic light states, since the colors will be more visible for the model to distinguish between. This observation can be applied to wherever the model is executed, whether in a small town such as Cumming, or a large city such as Atlanta.

To improve the timing at which the GUI was invoked, it was suggested in section 4.13 to increase the distance threshold from 100 to 200 m and eliminate the directional constraint. Upon making this change, the GUI was found to be invoked at a larger distance from the traffic light every time a signal was encountered in Day 3 testing; ranging from 100 to 150 m, as opposed to the mere 50 m in Day 2 testing. This is ample time for the driver to be able to read and react to the update offered to them while ensuring it is for the right traffic light on their route. The 200 m threshold should not be exceeded as the finding the nearest traffic lights segment (discussed in section 3.4.1) could point to the wrong traffic light at the same intersection as the closest to the driver, especially due to the lack of directional constraints. In that event, the updates delivered to

the driver would not be correct or usable for them to make a decision about their driving behavior.

The last portion of the integrated system to be evaluated in day 3 testing is the accuracy of the signal length predictions generated and relayed back to the driver. The latter can be determined by meticulously comparing the times at which the signal was encountered by the driver (Figure 48) to the predictions offered to them on the GUI (Table 10). As mentioned in section 3.4.4, detailing the functionality of the batch updates stored procedure and the knowledgebase, signal records representing the same traffic light state found within 10 seconds of one another will be appended to the same KB record. Signal records representing either a different traffic light state or found past the 10 second threshold of one another will be appended to a different KB record or create a new record. Using this ideology, the KB records presented in Figure 49 (cumulative up to day 3), can be validated, through comparison with the signal records uploaded during day 3 testing and the KB records presented in Figure 47 (cumulative up to day 2). For example, the time at which signal 1 was encountered on day 3 was 14:34:31, and comparing this with the knowledgebase shown in Figure 47, it can be seen that this time would not fall within the limits of any already existing records. Further, it is 18 seconds away from the closest KB record at 14:34:49, which is more than the 10 second threshold, qualifying it to be a separate KB record. Since the green phase starts at 14:34:41 and continues until 14:34:48 for this signal (Figures 71 - 72 for Day 3 Testing records), the knowledgebase shown in Figure 49 includes a separate record at this time with a duration of 7 seconds. On the other hand, the time at which signal 3 was encountered was 14:36:06, and since this is within three seconds of the closest KB record at 14:36:09, shown in Figure 47, the time can be appended to this record, widening the duration of the prediction. This is also shown in Figure 49. It can be determined

from this analysis that the signal length predictions are generated with high accuracy, due to the fact that they rely on data "learned" by the system and refine over time as more learning data is accumulated.

Relaying signal length predictions back to the driver also follows a particular methodology. To illustrate this concept, the times at which each traffic light was first encountered on Day 3 of testing has been displayed below in Figure 48. As can be seen from this list, not all times fall within existing KB record time periods, necessitating a future prediction to be offered at the current time. This was discussed in section 4.14.2, where a KB record within an hour of the current time can be offered as a prediction to the driver if they do not encounter the traffic light within an existing KB record at that time period. Combining this approach with the times presented in Figure 48, the predictions offered to the driver, shown in Table 10, can be validated. Revisiting the first example of crossing signal 1, it can be determined that there is no valid prediction that can be offered at that time since a KB record doesn't exist for it. In this event, a signal length prediction within an hour of the current time is offered, if it exists, so that the driver can make an educated decision on controlling their speed in approaching the traffic light. The closest record is at 14:34:49 in this case, and hence the signal length duration determined at that time is relayed to the driver on the GUI, as shown in Table 10. On the other hand, the time at which signal 2 was encountered was 14:35:50, and when comparing this with the knowledgebase shown in Figure 46, it is clear that the time falls within a valid KB record time period, at 14:35:47, and hence a current signal length prediction can be offered. This is relayed back to the driver on the GUI, as shown in Table 10. Through this analysis, it is clear that predictions are relayed back to the driver with high accuracy, since the algorithm does not deliver predictions based on data hours away from the current time period where the signal

phasing effect could be introduced. This phenomenon would cause signal lengths to be different throughout the day due to traffic flow conditions. Overall, the entire prediction algorithm works as intended, updating smoothly to widen already established signal length predictions or create new predictions based on new learning data, and relaying only useful predictions back to the driver.

4.16: Overall System Effectiveness and General Comments

The traffic light learning and prediction integrated system is overall quite effective, in obtaining learning data and developing signal length predictions from the data. There is however, one inefficiency that could arise in the learning system would be due to the directional headings determined and used for signal record uploads, as mentioned in section 3.4.1. The directional headings are calculated every time the current location of the driver is polled using the GPS receiver. While this is extremely useful when authorizing signal record uploads for traffic lights in straight line view of the vehicle, it is not as helpful for traffic lights on windy roads, where the vehicle is not in sight of the light. In these types of cases, the directional heading only matches the traffic light heading as the vehicle closely approaches the light, allowing for signal records to be uploaded only after the driver almost passes the light. This could cause the learning system to not detect the traffic light state properly due to being in close proximity to the light, which in turn would lead to creating wrong learning data. Although this scenario has not been dealt with in the current work, it is a great starting point for making future improvements to the system.

An additional factor to consider in the results is the number of false Green and false Red predictions. As can be seen from the data obtained, there weren't many incorrect traffic light state predictions recorded during the final tests of the integrated system, but amongst the

wrongly identified states, there were a couple of false Red and false Green predictions. In a way, false Red predictions are still fail-safe, because the worst that can happen is a driver coming to a halt at a red light, when the state is actually green and they have the clearance to go. This will not cause any major accidents, unlike when a false Green prediction is delivered, which would inform drivers to go through a light even when its state is Red and cause collisions with other vehicles travelling through the intersection. Although the results of this system were lucky enough to offer mostly false Red predictions, one way to address existing and future false Green predictions would be to incorporate confidence level (determined by the bounding boxes on the original live stream video capture frame, which were suppressed for the final demo) as an additional constraint in determining whether the state found by the model was accurate. If confidence levels are above 50% for all the detections found on the original video capture frame, signal records can be proceeded to be uploaded into the central learning database table, otherwise, the state determined should be disregarded as inaccurate. Although this would prevent traffic light state data from being acquired for some signals, it will ensure that improper learning data doesn't get added to the system, and thereby prevent inaccurate signal length predictions from being offered to the driver.

The last known state is included in the GUI update for every traffic light that has a signal record uploaded into the central learning database table prior to another car encountering it. While this may be considered as unnecessary information for signals that are adaptive or having varying phase schedules, the last known state's primary use is for signals in rural areas with very little to no signal record uploads made throughout the day. This will allow a driver that is encountering the signal to have at least an idea about its behavior at a particular time, so that they can decide to a reasonable extent on what speed to adopt when going through the light. For

traffic lights that are frequently encountered by drivers, the last known state update can be used in conjunction with the signal length predictions developed for the light in order to determine optimal speed profiles.

This section will now conclude with comments on the processing speed of the integrated traffic light learning and prediction system. There is a slight delay in uploading signal records into the central learning database table (approximately 1.5 seconds) after the model recognizes the traffic light state. There is also a minor delay in delivering driver updates on the GUI (approximately 2 seconds), due to having to invoke the easygui package within the Python script afresh every time the driver encounters a traffic light. Both of these delays lead to minor inaccuracies in developing signal length predictions, since the model can only perform detection after a signal record is uploaded once the first time of detection is complete. This may cause some state detections to not be performed by the model before the driver crosses the traffic light, shortening the overall signal length prediction due to not being able to upload signal records for those states. Future work can be done to improvise the system and reduce these delays, but, overall, the traffic light learning and prediction system operates at a reasonable speed, ensuring that signal records are uploaded and updates are delivered in a relatively timely manner to convenience the driver as they pass through each light on their route.

CHAPTER 5. FUTURE WORK & IMPROVEMENTS

This chapter provides thought for future work and improvements to the designed system presented in this document. Section 5.1 starts by suggesting predictive analytics using supervised machine learning techniques as a way to offer signal length predictions to drivers encountering traffic lights at time periods in which KB records do not exist. Section 5.2 talks about integrating other signal duration datasets, traffic updates, and weather advisories with the currently developed system. Section 5.3 continues to discuss how the model could be trained to achieve better performance by submitting a larger volume of images taken in different weather and lighting conditions to the neural network. Section 5.4 considers incorporating yellow light detection into a more improvised future learning system. Lastly, Section 5.5 discusses a more effective deployment strategy for the entire system, by combining the work done prior to this research with the currently developed system.

5.1: Predictive Analytics to Form Signal Length Predictions

Offering signal length predictions at every possible time that a driver encounters a traffic light can be challenging if the knowledgebase needs to rely on actual data. It would be extremely time-consuming and nearly impossible to have people drive through a traffic light at every single second throughout a day so that we could refine our knowledgebase to a granular level and offer exact signal length predictions. Supervised machine learning algorithms have been more recently explored to find a solution to such dilemmas, using already aggregated data to develop strong prediction models that can approximate data at an unknown time. This is the field of predictive analytics which is becoming more and more popular amongst companies across the world to identify technology trends and best practices. Using predictive analytics to offer signal length

predictions at a time when KB records don't actually exist would be a great addition and future improvement to the model, revolutionizing it to adapt to real traffic signal schedules. Figure 50 illustrates an example of using this technique to offer predictions to the driver.

Time From	Time To	Signal Duration Prediction	Traffic Light State
10:00	10:02	2 min	Green
10:46	10:47	1 min	Red
11:14	11:16	2 min	Green
11:28	11:29	1 min	Red
12:07	12:09	2 min	Green
12:36	12:37	1 min	Red
12:59	1:01	2 min	Green

Traffic Lights Knowledgebase

Current Time: 11:47 a.m.

Current State: Green

Prediction Offered to Driver:

Signal will remain green for approximately two mins.

Figure 50 - Predictive Analytics using Sample KB Records

Using the records surrounding the time frame at which the driver encounters the traffic light and observing the pattern of signal length durations formed at those times, a predictive analytics algorithm can determine a reasonable signal length prediction approximation for that time frame, even though no KB records exist for it. Eventually, employing a self-learning algorithm such as this would allow predictions to be generated with higher accuracy, especially if more "learning data" is submitted to the algorithm and knowledgebase.

5.2: Integrating Other Signal Duration Datasets, Traffic Updates, and Weather Advisories

In the background section of this document, signal data capture algorithms (such as Signal Guru, etc.) were mentioned as traffic light assistants that accumulated phase timings as drivers passed through the light. The data aggregated from these algorithms is equally reliable to the data captured by the currently developed system since both use live stream recognition to process the traffic light states found by the model. This allows only accurate traffic light state

data varied by timestamp to be accumulated into the central database. Merging the learning algorithm from the currently developed system with the data captured by the above-mentioned algorithms would allow for more data aggregation and aid in developing extremely accurate signal durations, hence making it a great addition to a future system.

In addition to integrating datasets, offering timely updates on traffic congestion, relative velocity and spacing with respect to other vehicles, and weather advisories is useful to a driver, primarily for safety concerns. In order to offer these types of updates to a driver, it is necessary to obtain a lot of information from the vehicle's environmental surroundings and integrate it with the on-board logistics and current developed traffic light learning and prediction system. Although the current system does not have scope to incorporate these features, a future system can surely accomplish this feat.

5.3: Training the Model to Achieve Higher Levels of Performance

As seen in the results section of the report (Chapter 4), both models do a fairly good job in detecting traffic lights in a few weather and lighting conditions; with Model 3 exceeding the performance of Model 2. The final selected traffic light recognition model, Model 3, was able to detect more than 90% of red and green traffic lights encountered in the live stream video feed during testing. However, there is still room for improvement in detection capability, especially when it comes to subjecting the model to more drastic weather and lighting conditions. Some weather conditions that are extremely prevalent in other parts of the country at different times of the year include blinding sunlight, heavy rain, dense fog, etc. To make a model that is adaptable in a more widespread nature, all of these conditions must be included in training. Future researchers should spend more time collecting a large volume of images in these different

weather conditions, so that the neural network can process the learning data to refine the model further. Model 3 is currently trained on approximately 2500 images; an industrial scale successful image recognition model should be trained on at least 10000 images to provide solid traffic light recognition with close to no misdetections. This is possible with a coordinated team of researchers investing time, money, and efforts to collect images and train the model.

5.4: Incorporating "Yellow Lights" into the Model

As discussed in section 3.3.2, the final model does not include yellow light configurations. This causes the integrated system's overall accuracy to drop, due to combining the durations of the red and yellow phases in developing predictions to relay back to the driver, which overestimates the red phase length. This was seen in the results shown in Tables 2, 3, 7, and 8, where a few yellow phases had been mis-detected as red phases, leading to inaccurate signal length determinations. To avoid this confusion, it is necessary to include the yellow light configuration into a further trained model, and separate out signal lengths by green phase, yellow phase, and red phase, when developing predictions. In order to add the yellow light configurations and ensure that proper detection capability in all types of weather conditions and environmental surroundings is established, it is necessary to submit at least 1000 more images of yellow lights to the neural network and train the model upon the existing framework until losses diminish below 2.00 (this was the loss limit set for the final model used in this assignment). This will create a more substantial traffic light recognition model that can be used in future automotive applications.

5.5: A Better Strategy for Deploying the Entire System

The currently designed system presented in this research is deployed using a laptop connected to a GPS Receiver and a Webcam, placed inside the vehicle. The entire system, while autonomously executed after the push of the run button in Python, shows all updates and uploads on the laptop screen and Python console respectively, making it hard for a driver to follow while paying attention to the road. To eliminate the use of a laptop and allow for easier execution of the entire system, the Android app developed in the prior work can be integrated with the current system. A Raspberry Pi can be used to replace the PC, by storing the code necessary to run the traffic light learning and prediction algorithms. The Pi camera module can replace the external web cam to allow for traffic light detection in live stream video feed. The Android App can be connected to the Raspberry Pi through Wi-Fi, and since the app already employs the Google Maps Distance Matrix API to obtain GPS data, updates will be able to be delivered to the driver on time before they approach a signal, as they are in the currently designed system. This will keep driver distraction to a minimum and make it easier overall to obtain and display real-time updates in a more compact way. Figure 51 shows a visual of the improvised system.

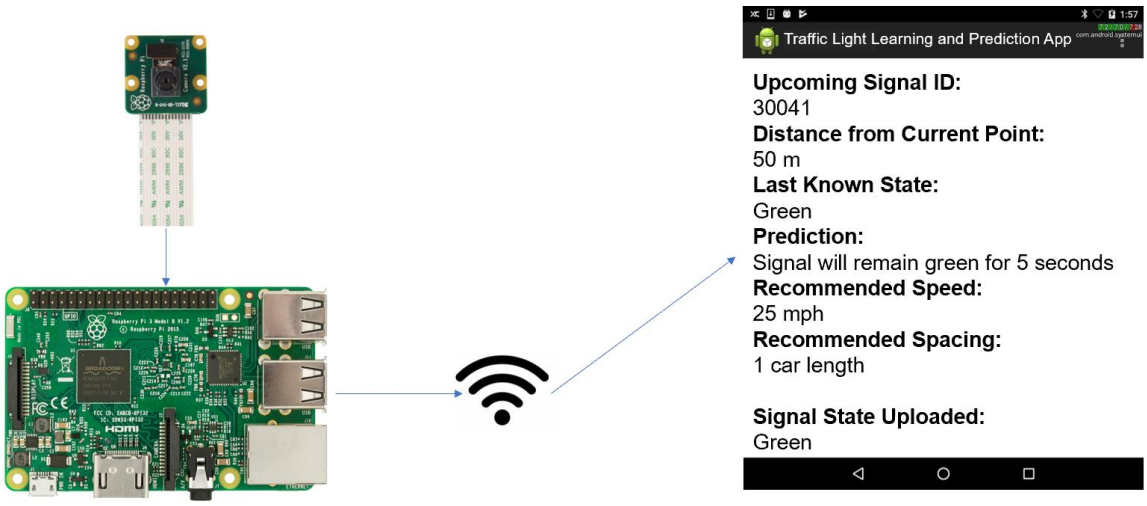

Figure 51 - Improvised System (using Raspberry Pi, Camera Module, Android App with Wi-Fi Connection)

CHAPTER 6. CONCLUSIONS

This chapter provides a conclusion to the topic presented in the paper, re-outlining the motivations to pursue research in this area and providing a summary of the accomplishments and shortcomings of the implemented research methodology. The chapter ends by presenting a few questions left open for thought at the end of this research assignment.

Technology has been soaring over the years, especially in the automotive industry, with the rise of V2I/V2V communications. The desire to obtain and use information necessary to predict traffic flow or commute time is gaining more interest as automobile companies look into fuel savings, energy consumption, environmental pollution, and vehicle part life-time. Further, on a user-end, reducing traffic congestion and the time to commute is in the interest of all hard-working professionals across the country. The ability to spend quality time with their families without having to sit through hours of traffic is a blessing that all workers would wish for.

To account for all of these factors, a traffic light learning and prediction system is designed in this research, with the ability to predict signal length durations based on "learned" traffic signal state data. The traffic light learning algorithm obtains "learning" data by running a traffic light recognition model on live stream video feed to identify and upload signal state detections by timestamp for each light. This model was meticulously developed using the Google TensorFlow API for object detection and was trained on over 2500 images of traffic lights in Atlanta and the northern Georgia area. The traffic light prediction algorithm uses the "learned" data, organizing it into records consisting of signal durations, using the NBUSP. It then stores these records into a knowledgebase that is contacted real-time during the drive to offer the driver updates on the predicted signal length of the upcoming traffic light on their route.

Overall, the traffic light learning and prediction system designed and outlined thoroughly in this document allows for successful traffic light state detection and signal duration determination. The computer vision system is able to detect, learn, and process captured signal states with over 80% accuracy and the prediction system is able to relay back inferences on signal length to the driver in a timely manner before they reach the traffic light. This saves drivers valuable time by preventing unnecessary waits at traffic lights. It also allows drivers to be informed ahead of time that they are approaching a traffic light, in the event that they are not in visible view of the light or distracted by other matters, preventing the total number of accidents on the road due to intersection collisions. Additionally, the driver will be alerted on when the state of the traffic light will change, so that they can either guide their speed through the light or at least slow down and stop at the light, turning off their engines to prevent idling. This would prevent unnecessary fuel wastage, environmental pollution and the costs associated with both of these factors for a driver and the general population of a city respectively. Further, executing a system as discussed in the future work section, that integrates the prior work in offering relative velocity and spacing recommendations along with other driver updates shown in the currently developed system, will enable a driver to improve their vehicle's energy efficiency and fuel economy parameters.

The system developed in this assignment also eliminates the need to obtain governmental access to traffic light phase timings or pay extremely expensive subscriptions to car companies in order to have access to signal length predictions. It is an inexpensive solution that any drivers with ADAS systems (camera with capability to run traffic light recognition model on live stream video feed) can use to contribute to a centrally maintained learning database and refine signal length predictions. This concept can also be used as the foundation for predictive analysis, in

developing signal length predictions at times in which records do not exist within the knowledgebase, due to its widespread nature.

A few questions that are left open at the end of this project are regarding the scalability and implementation of the system designed. How would this system be scalable across vehicles? What compact solution will replace laptops in vehicles while allowing drivers to still contribute to the learning and prediction algorithms and receive updates real-time? What are the tradeoffs between cost and scalability in designing a more compact solution? All of these questions are left to be answered by future researchers working on improving and implementing this system.

APPENDIX

SECTION A. Traffic Light State Recognition Model 1: Detection Using Color-Based Differentiation and Traffic Light Shape Separately

The first model developed differentiates traffic lights primarily by color, i.e. it treats a green circle as a green light, a red circle as a red light, etc., as shown in Figure 12. While model 1 was able to perform detections on roads were there weren't any distractions such as street signs with lights, it had difficulty in detecting traffic lights in very congested roads since it confused the lights with a car's tail-lights and other objects. It was deemed from analyzing the results that Model 1's detection capability was impacted negatively because it used just color-based differentiation to identify traffic lights, making it hard to uniquely identify the light's shape from other similar looking objects. Due to its lower performance capability compared to Models 2 and 3, discussed in sections 3.3.1 and 3.3.2, it was determined not to use Model 1 for any final model selection tests or the integrated system.

SECTION B. Integrating the System

B.1: The Multiprocessing and Threading Packages in Python

In Python, the multiprocessing package allows users to run two functions at once. This could be two small functions such as adding and subtracting numbers, or two big functions with while and for loops performing operations on the provided inputs. Since the traffic light learning system utilizes infinite while loops to perform live stream video recognition and poll GPS data every second, the multiprocessing package was used to run the loops in parallel by placing each loop in its own function.

As can be seen from the figure, a main function definition was created to start the first (GPS polling and driver updates) and second (live-stream video recognition) processes and join them so they occur at once using the multiprocess.Process module. The Queue module is also imported in this function and several variables are initialized as queues and fed as inputs to the main two functions. In multiprocessing, global variables are not recognized across different functions, and hence queues are necessary in order for the variable values to be transferred. In the case of this project, three variables (distance to the closest light, timestamp of detection, and traffic light state detected) must be globalized and used across functions, and hence are placed within queues.

Although the multiprocessing package allows the two loops to be run at once so that updates are fed to the driver while capturing live stream traffic light detections, the queues used to transfer variable values do not execute smoothly and cause runtime error due to the large amounts of data that get appended to it during each iteration. The flaws involved in this approach will be presented in the results and discussion section of the proposal with testing data to support observations.

After multiprocessing was tested and deemed as an insufficient approach to integrating the system, the threading package in Python was implemented as a new solution. While this is coded in a very similar way to multiprocessing, requiring each loop to be in a separate function, threading involves less memory overhead and has the ability to intercommunicate better within functions, therefore reducing the need to have queues to process global variable data. The drawback to the threading package is that while both threads start at once, only the first thread is executed until its condition is satisfied. Although this is counter-intuitive, it is due to the lack of

a join function to run both threads simultaneously, making the threading package an obsolete solution for Python users. This was realized after a few runs of testing, the results of which will be discussed in Chapter 4.

B.2: The Database Read/Write Method

The main issue involved with integrating the system was in transferring the values of the global variables when the two functions are executed simultaneously. While multiprocessing and threading dealt with that component effectively, they had problems with data storage and simultaneous execution, respectively, which made them ineffective methods to integrate the system when used on their own. However, since the multiprocessing package still has the ability to run both loops in parallel, another solution having to integrate the system in a multiprocessing structure with databases to store global variable data was developed. This solution allowed data that needed to pass from one function to be uploaded and stored in temporary database tables, which were automatically cleared as soon as the data was read into a local variable in the other function. These database tables were stored directly in memory (RAM) instead of on disk for fast retrieval of data.

The problem involved in this approach was that multiprocessing treats database storage in one function and retrieval in another as global variables. Therefore, although data can be stored from one function into a table, it cannot be retrieved from another function, since it's not a value local to the function. More detail on the process involved in testing this approach and the corresponding results will be discussed in Chapter 4. After all the three above-mentioned approaches were tested, it was determined that the only option to integrating the system was to

avoid any form of multiprocessing or simultaneous loop execution; instead performing continuous loop integration to execute both functions one after another.

SECTION C: Model Configurations

C.1: Model 2 Configurations

Figure 52 - Green Light (Model 2)

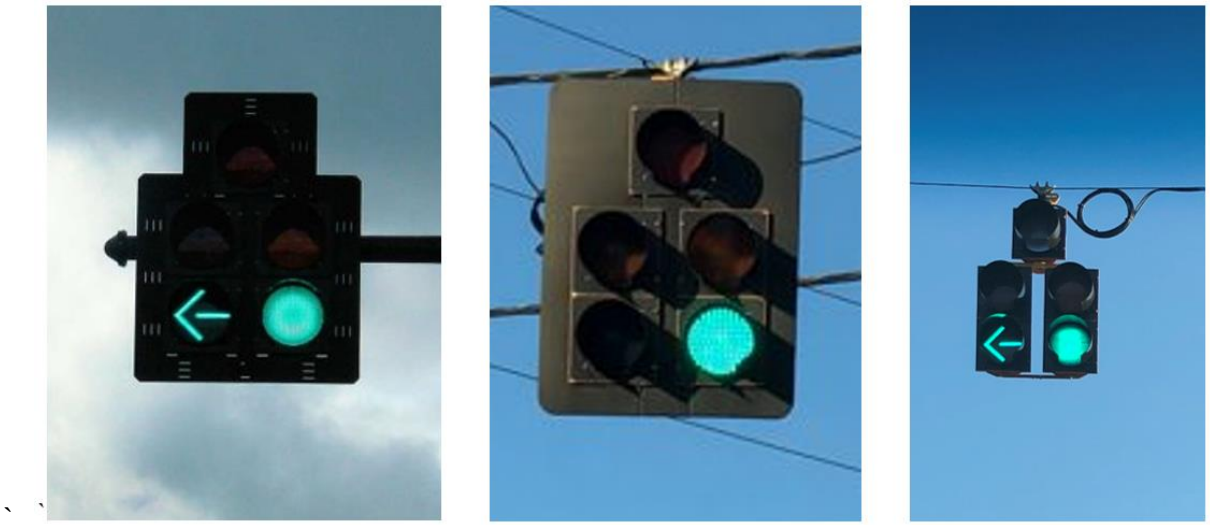

Figure 53 - Green Light 2 (Model 2)

Figure 54 - Green Right Arrow (Model 2)

Figure 55 - Green Left Arrow (Model 2)

Figure 56 - Red Left Arrow (Model 2)

Figure 57 - Red Light (Model 2)

Figure 58 - Red Light 2 (Model 2)

C.2: Model 3 Configurations

Figure 59 - Green Light (Model 3)

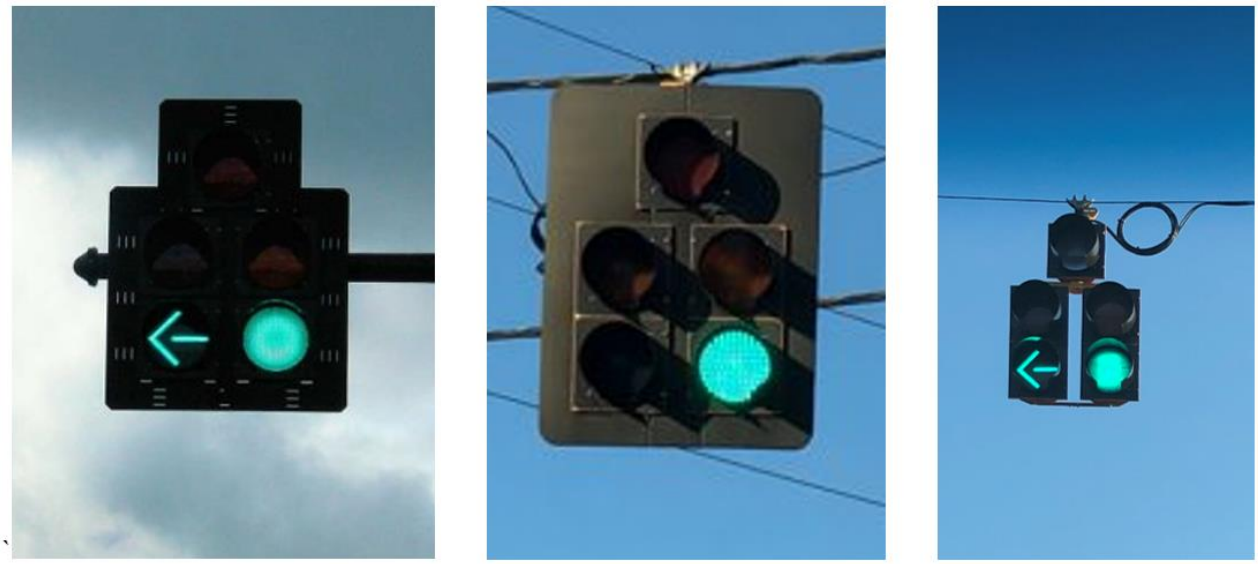

Figure 60 - Green Light 2 (Model 3)

Figure 61 - Green Light 3 (Model 3)

Figure 62 - Green Right Arrow (Model 3)

Figure 63 - Red Light (Model 3)

Figure 64 - Red Light 2 (Model 3)

Figure 65 - Red Light 3 (Model 3)

SECTION D. Integrated System Testing Data (Days 1 – 3)

D.1: Day 1 Testing Data

signalId	datetime	state	300419	14:36:34	Green	3004111	14:38:20	Green	3004113	14:39:56	Green
300418	14:35:49	Red	300419	14:36:35	Green	3004111	14:38:20	Green	3004113	14:40:00	Green
300418	14:35:51	Red	300419	14:36:36	Green	3004111	14:38:21	Green	3004113	14:40:01	Green
300418	14:35:52	Red	300419	14:36:37	Green	3004111	14:38:22	Green	3004113	14:40:02	Green
300418	14:35:52	Red	300419	14:36:38	Green	3004111	14:38:23	Green	3004113	14:40:02	Green
300418	14:35:53	Red	300419	14:36:38	Green	3004111	14:38:23	Green	3004113	14:40:03	Green
300418	14:35:54	Red	300419	14:36:39	Green	3004111	14:38:25	Green	3004113	14:40:04	Green
300418	14:35:55	Red	300419	14:36:40	Green	3004111	14:38:26	Green	3004113	14:40:05	Green
300418	14:35:55	Red	300419	14:36:40	Green	3004112	14:39:15	Green	3004113	14:40:05	Green
300418	14:35:56	Green	300419	14:36:41	Green	3004112	14:39:16	Green	3004113	14:40:06	Green
300418	14:35:57	Green	300419	14:36:42	Green	3004112	14:39:16	Green	3004113	14:40:07	Green
300418	14:35:57	Green	300419	14:36:42	Green	3004112	14:39:17	Green	3004113	14:40:08	Green
300418	14:35:58	Green	300419	14:36:43	Green	3004112	14:39:18	Green	3004113	14:40:08	Green
300418	14:35:59	Green	300419	14:36:44	Green	3004112	14:39:19	Green	3004113	14:40:09	Green
300418	14:35:59	Green	300419	14:36:44	Green	3004112	14:39:19	Green	3004113	14:40:10	Green
300418	14:36:00	Green	300419	14:36:45	Green	3004112	14:39:20	Green	3004113	14:40:11	Green
300418	14:36:01	Green	300419	14:36:46	Green	3004112	14:39:21	Green	3004113	14:40:11	Green
300418	14:36:01	Green	300419	14:36:46	Green	3004112	14:39:22	Green	3004113	14:40:12	Green
300418	14:36:02	Green	300419	14:36:47	Green	3004112	14:39:22	Green	3004113	14:40:13	Green
300419	14:36:24	Green	3004110	14:37:01	Green	3004112	14:39:23	Green	3004113	14:40:13	Green
300419	14:36:25	Green	3004110	14:37:02	Green	3004112	14:39:24	Green	3004113	14:40:14	Green
300419	14:36:26	Green	3004110	14:37:03	Green	3004112	14:39:25	Green	3004113	14:40:15	Green
300419	14:36:27	Green	3004110	14:37:04	Green	3004112	14:39:25	Green	3004113	14:40:16	Green
300419	14:36:29	Green	3004110	14:37:04	Green	3004112	14:39:26	Green	3004113	14:40:16	Green
300419	14:36:29	Green	3004110	14:37:05	Green	3004112	14:39:27	Green	3004113	14:40:17	Green
300419	14:36:30	Green	3004110	14:37:06	Green	3004113	14:39:48	Green	3004113	14:40:18	Green
300419	14:36:31	Green	3004110	14:37:07	Green	3004113	14:39:50	Green	3004113	14:40:19	Green
300419	14:36:32	Green	3004111	14:38:17	Green	3004113	14:39:52	Green	3004113	14:40:20	Green
300419	14:36:33	Green	3004111	14:38:18	Green	3004113	14:39:54	Green	3004113	14:40:21	Green
300419	14:36:34	Green	3004111	14:38:19	Green	3004113	14:39:55	Green	3004113	14:40:22	Green

Figure 66 - Integrated System Day 1 Testing Data (1)

3004113	14:40:22	Green	3004113	14:40:47	Green	3004120	14:43:13	Green	3004121	14:45:10	Green
3004113	14:40:23	Green	3004113	14:40:48	Green	3004120	14:43:14	Green	3004121	14:45:11	Green
3004113	14:40:24	Green	3004113	14:40:49	Green	3004120	14:43:14	Green	3004121	14:45:12	Green
3004113	14:40:25	Green	3004113	14:40:50	Red	3004120	14:43:15	Green	3004121	14:45:12	Green
3004113	14:40:26	Green	3004113	14:40:51	Red	3004120	14:43:16	Green	3004121	14:45:13	Green
3004113	14:40:27	Green	3004113	14:40:51	Red	3004120	14:43:17	Green	3004121	14:45:14	Green
3004113	14:40:28	Green	3004113	14:40:52	Red	3004120	14:43:17	Green	3004122	14:45:27	Green
3004113	14:40:28	Green	3004113	14:40:53	Red	3004120	14:43:18	Green	3004122	14:45:28	Green
3004113	14:40:29	Green	3004113	14:40:54	Red	3004120	14:43:19	Green	3004122	14:45:28	Green
3004113	14:40:30	Green	3004113	14:40:55	Red	3004120	14:43:20	Green	3004122	14:45:29	Green
3004113	14:40:31	Green	3004113	14:40:56	Red	3004120	14:43:20	Green	3004122	14:45:30	Green
3004113	14:40:31	Green	3004113	14:40:57	Red	3004120	14:43:21	Green	3004122	14:45:31	Green
3004113	14:40:32	Green	3004113	14:40:57	Red	3004120	14:43:22	Green	3004122	14:45:31	Green
3004113	14:40:33	Green	3004119	14:42:02	Green	3004120	14:43:23	Green	3004122	14:45:32	Green
3004113	14:40:34	Green	3004119	14:42:03	Green	3004121	14:44:58	Green	3004122	14:45:33	Green
3004113	14:40:34	Green	3004119	14:42:04	Green	3004121	14:44:59	Green	3004122	14:45:33	Green
3004113	14:40:35	Green	3004119	14:42:04	Green	3004121	14:45:00	Green	3004122	14:45:34	Green
3004113	14:40:36	Green	3004119	14:42:05	Green	3004121	14:45:00	Green	3004122	14:45:35	Green
3004113	14:40:37	Green	3004119	14:42:06	Green	3004121	14:45:01	Green	3004123	14:45:58	Green
3004113	14:40:38	Green	3004119	14:42:07	Green	3004121	14:45:02	Green	3004123	14:45:58	Green
3004113	14:40:38	Green	3004119	14:42:08	Green	3004121	14:45:03	Green	3004123	14:46:00	Green
3004113	14:40:39	Green	3004119	14:42:08	Green	3004121	14:45:04	Green	3004123	14:46:00	Green
3004113	14:40:40	Green	3004119	14:42:09	Green	3004121	14:45:04	Green	3004123	14:46:02	Green
3004113	14:40:41	Green	3004119	14:42:10	Green	3004121	14:45:05	Green	3004123	14:46:02	Green
3004113	14:40:42	Green	3004119	14:42:11	Green	3004121	14:45:06	Green	3004123	14:46:03	Green
3004113	14:40:43	Green	3004119	14:42:12	Green	3004121	14:45:07	Green	3004123	14:46:04	Green
3004113	14:40:44	Green	3004120	14:43:10	Green	3004121	14:45:07	Green	3004123	14:46:05	Green
3004113	14:40:44	Green	3004120	14:43:11	Green	3004121	14:45:08	Green	3004123	14:46:06	Green
3004113	14:40:45	Green	3004120	14:43:11	Green	3004121	14:45:09	Green	3004123	14:46:06	Green
3004113	14:40:46	Green	3004120	14:43:12	Green	3004121	14:45:10	Green	3004123	14:46:07	Green

Figure 67 - Integrated System Day 1 Testing Data (2)

D.2: Day 2 Testing Data

signalId	datetime	state	300419	14:35:36	Red	3004110	14:36:12	Green	3004112	14:38:23	Green
300418	14:34:49	Green	300419	14:35:37	Red	3004110	14:36:13	Green	3004112	14:38:24	Green
300418	14:34:53	Green	300419	14:35:38	Red	3004110	14:36:14	Green	3004112	14:38:24	Green
300418	14:34:54	Green	300419	14:35:38	Red	3004110	14:36:14	Green	3004112	14:38:25	Green
300418	14:34:55	Green	300419	14:35:39	Red	3004110	14:36:15	Green	3004113	14:38:36	Red
300418	14:34:56	Green	300419	14:35:40	Red	3004111	14:37:20	Green	3004113	14:38:38	Red
300418	14:34:57	Green	300419	14:35:40	Red	3004111	14:37:21	Green	3004113	14:38:36	Red
300418	14:34:58	Green	300419	14:35:41	Red	3004111	14:37:22	Green	3004113	14:38:38	Red
300418	14:34:59	Green	300419	14:35:42	Red	3004111	14:37:23	Green	3004113	14:38:36	Red
300418	14:35:00	Green	300419	14:35:42	Red	3004111	14:37:24	Green	3004113	14:38:38	Red
300418	14:35:00	Green	300419	14:35:43	Red	3004111	14:37:24	Green	3004113	14:38:36	Red
300418	14:35:01	Green	300419	14:35:44	Red	3004111	14:37:25	Green	3004113	14:38:38	Red
300419	14:35:22	Red	300419	14:35:44	Red	3004111	14:37:26	Green	3004113	14:38:36	Red
300419	14:35:23	Red	300419	14:35:45	Red	3004111	14:37:26	Green	3004113	14:38:38	Red
300419	14:35:24	Red	300419	14:35:46	Red	3004111	14:37:27	Green	3004113	14:38:36	Red
300419	14:35:25	Red	300419	14:35:47	Red	3004111	14:37:27	Green	3004113	14:38:38	Red
300419	14:35:26	Red	300419	14:35:47	Green	3004111	14:37:28	Green	3004113	14:38:36	Red
300419	14:35:26	Red	300419	14:35:48	Green	3004111	14:37:29	Green	3004113	14:38:38	Red
300419	14:35:27	Red	300419	14:35:49	Green	3004112	14:38:15	Green	3004113	14:38:36	Red
300419	14:35:28	Red	300419	14:35:49	Green	3004112	14:38:16	Green	3004113	14:38:38	Red
300419	14:35:28	Red	300419	14:35:50	Green	3004112	14:38:17	Green	3004113	14:38:36	Red
300419	14:35:29	Red	300419	14:35:51	Green	3004112	14:38:18	Green	3004113	14:38:38	Red
300419	14:35:30	Red	300419	14:35:51	Green	3004112	14:38:18	Green	3004113	14:38:39	Red
300419	14:35:31	Red	300419	14:35:52	Green	3004112	14:38:19	Green	3004113	14:38:40	Red
300419	14:35:31	Red	300419	14:35:53	Green	3004112	14:38:19	Green	3004113	14:38:40	Red
300419	14:35:32	Red	300419	14:35:54	Green	3004112	14:38:20	Green	3004113	14:38:41	Red
300419	14:35:33	Red	300419	14:35:54	Green	3004112	14:38:21	Green	3004113	14:38:42	Red
300419	14:35:34	Red	300419	14:35:55	Green	3004112	14:38:21	Green	3004113	14:38:43	Red
300419	14:35:35	Red	300419	14:35:56	Green	3004112	14:38:22	Green	3004113	14:38:44	Red
300419	14:35:35	Red	3004110	14:36:09	Green	3004112	14:38:23	Green	3004113	14:38:45	Red

Figure 68 - Integrated System Day 2 Testing Data (1)

3004113	14:38:46	Red	3004113	14:39:13	Red	3004113	14:39:44	Green	3004120	14:41:08	Green
3004113	14:38:47	Red	3004113	14:39:14	Red	3004113	14:39:45	Green	3004120	14:41:09	Green
3004113	14:38:48	Red	3004113	14:39:15	Red	3004119	14:40:00	Green	3004120	14:41:09	Green
3004113	14:38:48	Red	3004113	14:39:15	Red	3004119	14:40:01	Green	3004121	14:42:15	Green
3004113	14:38:49	Red	3004113	14:39:13	Red	3004119	14:40:02	Green	3004121	14:42:16	Green
3004113	14:38:50	Red	3004113	14:39:14	Red	3004119	14:40:03	Green	3004121	14:42:17	Green
3004113	14:38:51	Red	3004113	14:39:15	Red	3004119	14:40:04	Green	3004121	14:42:18	Green
3004113	14:38:51	Red	3004113	14:39:15	Red	3004119	14:40:04	Green	3004121	14:42:18	Green
3004113	14:38:52	Red	3004113	14:39:17	Red	3004119	14:40:05	Green	3004121	14:42:19	Green
3004113	14:38:53	Red	3004113	14:39:18	Red	3004119	14:40:06	Green	3004121	14:42:20	Green
3004113	14:38:54	Red	3004113	14:39:19	Red	3004119	14:40:06	Green	3004121	14:42:20	Green
3004113	14:38:55	Red	3004113	14:39:21	Red	3004119	14:40:07	Green	3004121	14:42:21	Green
3004113	14:38:56	Red	3004113	14:39:23	Red	3004119	14:40:08	Green	3004121	14:42:21	Green
3004113	14:38:57	Red	3004113	14:39:25	Red	3004119	14:40:09	Green	3004121	14:42:22	Green
3004113	14:38:58	Red	3004113	14:39:26	Red	3004119	14:40:09	Green	3004121	14:42:23	Green
3004113	14:38:59	Red	3004113	14:39:27	Red	3004119	14:40:10	Green	3004121	14:42:23	Green
3004113	14:39:01	Red	3004113	14:39:28	Red	3004120	14:40:59	Green	3004121	14:42:24	Green
3004113	14:39:01	Red	3004113	14:39:29	Red	3004120	14:41:00	Green	3004121	14:42:24	Green
3004113	14:39:02	Red	3004113	14:39:32	Red	3004120	14:41:01	Green	3004121	14:42:25	Green
3004113	14:39:03	Red	3004113	14:39:33	Red	3004120	14:41:02	Green	3004122	14:42:36	Red
3004113	14:39:04	Red	3004113	14:39:34	Red	3004120	14:41:02	Green	3004122	14:42:37	Red
3004113	14:39:05	Red	3004113	14:39:35	Red	3004120	14:41:03	Green	3004122	14:42:39	Red
3004113	14:39:06	Red	3004113	14:39:36	Red	3004120	14:41:03	Green	3004122	14:42:40	Red
3004113	14:39:06	Red	3004113	14:39:37	Red	3004120	14:41:04	Green	3004122	14:42:40	Red
3004113	14:39:07	Red	3004113	14:39:38	Red	3004120	14:41:05	Green	3004122	14:42:41	Red
3004113	14:39:09	Red	3004113	14:39:40	Red	3004120	14:41:05	Green	3004122	14:42:41	Red
3004113	14:39:10	Red	3004113	14:39:41	Red	3004120	14:41:06	Green	3004122	14:42:42	Red
3004113	14:39:10	Red	3004113	14:39:42	Green	3004120	14:41:06	Green	3004122	14:42:43	Red
3004113	14:39:11	Red	3004113	14:39:42	Green	3004120	14:41:07	Green	3004122	14:42:43	Red
3004113	14:39:12	Red	3004113	14:39:43	Green	3004120	14:41:07	Green	3004122	14:42:44	Red

Figure 69 - Integrated System Day 2 Testing Data (2)

3004122	14:42:44	Red	3004122	14:43:10	Red	3004122	14:43:33	Green
3004122	14:42:45	Red	3004122	14:43:11	Red	3004122	14:43:34	Green
3004122	14:42:46	Red	3004122	14:43:12	Red	3004122	14:43:35	Green
3004122	14:42:47	Red	3004122	14:43:13	Red	3004122	14:43:35	Green
3004122	14:42:47	Red	3004122	14:43:14	Red	3004122	14:43:36	Green
3004122	14:42:48	Red	3004122	14:43:14	Red	3004122	14:43:36	Green
3004122	14:42:49	Red	3004122	14:43:15	Red	3004122	14:43:37	Green
3004122	14:42:49	Red	3004122	14:43:16	Red	3004122	14:43:37	Green
3004122	14:42:50	Red	3004122	14:43:17	Red	3004122	14:43:38	Green
3004122	14:42:51	Red	3004122	14:43:18	Red	3004122	14:43:39	Green
3004122	14:42:52	Red	3004122	14:43:20	Green	3004122	14:43:39	Green
3004122	14:42:53	Red	3004122	14:43:20	Green	3004122	14:43:40	Green
3004122	14:42:54	Red	3004122	14:43:21	Green	3004122	14:43:40	Green
3004122	14:42:55	Red	3004122	14:43:22	Green	3004123	14:43:59	Green
3004122	14:42:56	Red	3004122	14:43:22	Green	3004123	14:44:01	Green
3004122	14:42:57	Red	3004122	14:43:23	Green	3004123	14:44:02	Green
3004122	14:42:57	Red	3004122	14:43:24	Green	3004123	14:44:02	Green
3004122	14:42:58	Red	3004122	14:43:24	Green	3004123	14:44:03	Green
3004122	14:42:59	Red	3004122	14:43:25	Green	3004123	14:44:04	Green
3004122	14:43:00	Red	3004122	14:43:26	Green	3004123	14:44:05	Green
3004122	14:43:01	Red	3004122	14:43:26	Green	3004123	14:44:06	Green
3004122	14:43:02	Red	3004122	14:43:27	Green	3004123	14:44:07	Green
3004122	14:43:03	Red	3004122	14:43:28	Green	3004123	14:44:07	Green
3004122	14:43:04	Red	3004122	14:43:29	Green	3004123	14:44:08	Red
3004122	14:43:04	Red	3004122	14:43:29	Green	3004123	14:44:12	Red
3004122	14:43:05	Red	3004122	14:43:30	Green			
3004122	14:43:06	Red	3004122	14:43:31	Green			
3004122	14:43:07	Red	3004122	14:43:31	Green			
3004122	14:43:08	Red	3004122	14:43:32	Green			
3004122	14:43:09	Red	3004122	14:43:33	Green			

Figure 70 - Integrated System Day 2 Testing Data (3)

D.3: Day 3 Testing Data

signalId	datetime	state	3004110	14:36:09	Green	3004119	14:40:36	Green	3004120	14:42:18	Red
300418	14:34:31	Green	3004110	14:36:10	Green	3004119	14:40:36	Green	3004120	14:42:19	Red
300418	14:34:32	Green	3004110	14:36:11	Green	3004119	14:40:36	Green	3004120	14:42:20	Red
300418	14:34:33	Green	3004110	14:36:12	Green	3004119	14:40:37	Green	3004120	14:42:21	Red
300418	14:34:34	Green	3004110	14:36:13	Green	3004119	14:40:38	Green	3004120	14:42:22	Red
300418	14:34:35	Green	3004111	14:37:22	Green	3004119	14:40:39	Green	3004120	14:42:23	Red
300418	14:34:36	Green	3004111	14:37:23	Green	3004119	14:40:40	Green	3004120	14:42:24	Red
300418	14:34:36	Green	3004111	14:37:24	Green	3004119	14:40:41	Green	3004120	14:42:25	Red
300418	14:34:38	Green	3004111	14:37:25	Green	3004119	14:40:42	Green	3004120	14:42:26	Red
300418	14:34:39	Green	3004111	14:37:26	Green	3004119	14:40:43	Green	3004120	14:42:27	Red
300418	14:34:40	Green	3004111	14:37:27	Green	3004119	14:40:44	Green	3004120	14:42:28	Red
300418	14:34:41	Green	3004111	14:37:28	Green	3004119	14:40:45	Green	3004120	14:42:29	Red
300418	14:34:42	Green	3004111	14:37:29	Green	3004119	14:40:45	Green	3004120	14:42:30	Red
300418	14:34:43	Green	3004111	14:37:30	Green	3004119	14:40:46	Green	3004120	14:42:31	Red
300418	14:34:44	Green	3004112	14:38:19	Green	3004120	14:42:05	Red	3004120	14:42:32	Red
300418	14:34:45	Green	3004112	14:38:20	Green	3004120	14:42:05	Red	3004120	14:42:33	Red
300418	14:34:45	Green	3004112	14:38:21	Green	3004120	14:42:06	Red	3004120	14:42:34	Red
300418	14:34:46	Green	3004112	14:38:22	Green	3004120	14:42:07	Red	3004120	14:42:35	Red
300418	14:34:46	Green	3004112	14:38:23	Green	3004120	14:42:07	Red	3004120	14:42:36	Red
300419	14:35:50	Green	3004112	14:38:24	Green	3004120	14:42:08	Red	3004120	14:42:37	Red
300419	14:35:51	Green	3004112	14:38:25	Green	3004120	14:42:09	Red	3004120	14:42:38	Red
300419	14:35:52	Green	3004112	14:38:26	Green	3004120	14:42:10	Red	3004120	14:42:39	Red
300419	14:35:53	Green	3004112	14:38:27	Green	3004120	14:42:11	Red	3004120	14:42:40	Red
300419	14:35:54	Green	3004112	14:38:28	Green	3004120	14:42:12	Red	3004120	14:42:41	Red
300419	14:35:55	Green	3004112	14:38:29	Green	3004120	14:42:13	Red	3004120	14:42:42	Red
300419	14:35:56	Green	3004112	14:38:30	Green	3004120	14:42:14	Red	3004120	14:42:42	Red
300419	14:35:57	Green	3004112	14:38:31	Green	3004120	14:42:15	Red	3004120	14:42:43	Red
3004110	14:36:06	Green	3004112	14:38:32	Green	3004120	14:42:16	Red	3004120	14:42:44	Red
3004110	14:36:07	Green	3004112	14:38:33	Green	3004120	14:42:16	Red	3004120	14:42:45	Red
3004110	14:36:08	Green	3004119	14:40:35	Green	3004120	14:42:17	Red	3004120	14:42:45	Red

Figure 71 - Integrated System Day 3 Testing Data (1)

3004120	14:42:45	Red	3004122	14:45:31	Green
3004120	14:42:46	Red	3004122	14:45:32	Green
3004120	14:42:47	Red	3004122	14:45:33	Green
3004120	14:42:48	Red	3004122	14:45:34	Green
3004120	14:42:49	Red	3004122	14:45:35	Green
3004120	14:42:50	Red	3004122	14:45:36	Green
3004120	14:42:50	Red	3004122	14:45:37	Green
3004120	14:42:51	Red	3004122	14:45:38	Green
3004120	14:42:52	Red	3004122	14:45:39	Green
3004120	14:42:52	Red	3004122	14:45:39	Green
3004120	14:42:53	Green	3004122	14:45:40	Green
3004120	14:42:54	Green	3004122	14:45:41	Green
3004120	14:42:55	Green	3004122	14:45:42	Green
3004120	14:42:56	Green	3004123	14:46:06	Green
3004120	14:42:56	Green	3004123	14:46:07	Green
3004120	14:42:57	Green	3004123	14:46:08	Green
3004121	14:43:51	Green	3004123	14:46:09	Green
3004121	14:43:52	Green	3004123	14:46:10	Green
3004121	14:43:53	Green	3004123	14:46:11	Green
3004121	14:43:53	Green	3004123	14:46:12	Green
3004121	14:43:54	Green	3004123	14:46:13	Green
3004121	14:43:55	Green			
3004121	14:43:55	Green			
3004121	14:43:56	Green			
3004121	14:43:57	Green			
3004121	14:43:57	Green			
3004121	14:43:58	Green			
3004121	14:43:58	Green			
3004121	14:43:58	Green			
3004121	14:43:59	Green			

Figure 72 - Integrated System Day 3 Testing Data (2)

REFERENCES

1. (2015). "Here's How Much Time Americans Waste in Traffic."

2. (2018) Traffic Technology Services, Inc. Establishes Oregon DOT as First Statewide Vehicle-to-Infrastructure Service, Escalates Virginia DOT to Largest.

3. (2019, 2018). "Statistics on Intersection Accidents." Retrieved September 2, 2019, 2019, from https://www.autoaccident.com/statistics-on-intersection-accidents.html.

4. Alkiek, K. (2018). "Identifying Regions of Interest." Traffic Light Recognition — A Visual Guide https://medium.com/@kenan.r.alkiek/https-medium-com-kenan-r-alkiek-traffic-light-recognition-505d6ab913b1 Accessed February 9, 2019 2019.

5. Almeida, T., et al. (2018). "Prototyping a Traffic Light Recognition Device with Expert Knowledge." Information **9**(11): 278.

6. Asadi, B. and A. Vahidi (2009). "Predictive use of traffic signal state for fuel saving." IFAC Proceedings Volumes **42**(15): 484-489.

7. Axer, S. and B. Friedrich (2016). "A methodology for signal timing estimation based on low frequency floating car data: Analysis of needed sample sizes and influencing factors." Transportation research procedia **15**: 220-232.

8. Bento, L. C., et al. (2012). Intelligent traffic management at intersections supported by V2V and V2I communications. 2012 15th International IEEE Conference on Intelligent Transportation Systems, IEEE.

9. De Charette, R. and F. Nashashibi (2009). Traffic light recognition using image processing compared to learning processes. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE.

10. De Nunzio, G., et al. (2016). "Eco- driving in urban traffic networks using traffic signals information." International Journal of Robust and Nonlinear Control **26**(6): 1307-1324.

11. Djahel, S., et al. (2015). Toward V2I communication technology-based solution for reducing road traffic congestion in smart cities. 2015 International Symposium on Networks, computers and communications (ISNCC), IEEE.

12. Estrada, Z. (2018) Now your Audi can read Washington, DC's traffic lights. Now your Audi can read Washington, DC's traffic lights

13. Fairfield, N. and C. Urmson (2011). Traffic light mapping and detection. 2011 IEEE International Conference on Robotics and Automation, IEEE.

14. Fayazi, S. A. and A. Vahidi (2015). "Crowdsourcing phase and timing of pre-timed

traffic signals in the presence of queues: algorithms and back-end system architecture." IEEE Transactions on Intelligent Transportation Systems **17**(3): 870-881.

15. Fayazi, S. A., et al. (2014). "Traffic signal phase and timing estimation from low-frequency transit bus data." IEEE Transactions on Intelligent Transportation Systems **16**(1): 19-28.

16. Gauthier, C. (2016) The Audi Red Light Countdown Clock Gamechanger. The Audi Red Light Countdown Clock Gamechanger **2019**,

17. Geiger, J. (2018) The Light Will Turn Green in 3, 2, 1: Audi Expands Traffic Signal Info. The Light Will Turn Green in 3, 2, 1: Audi Expands Traffic Signal Info

18. Herman, M. M. (2019). VDOT, Audi, and TTS Bring Traffic Light Information Technology to Virginia. VDOT maintained traffic signal network now communicating with select Audi vehicles in the Northern Virginia area, VDOT.

19. Howard, B. (2016) Hands on with Audi's exciting (no, really) traffic light countdown timer. Hands on with Audi's exciting (no, really) traffic light countdown timer

20. Ivanovic, I. (2018) Predicting traffic light behavior to smooth out traffic in a smart city. Predicting traffic light behavior to smooth out traffic in a smart city

21. Katsaros, K., et al. (2011). "Application of vehicular communications for improving the efficiency of traffic in urban areas." Wireless Communications and Mobile Computing **11**(12): 1657-1667.

22. Koukoumidis, E., et al. (2011). SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advisory. Proceedings of the 9th international conference on Mobile systems, applications, and services, ACM.

23. Krause, M. and K. Bengler (2013). My Phone, My Car and I-And Maybe a Traffic Light Assistant. ICONS 2013, The Eighth International Conference on Systems.

24. Lawitzky, A., et al. (2013). Energy optimal control to approach traffic lights. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE.

25. Li, L.-h., et al. (2018). "A Separation Strategy for Connected and Automated Vehicles: Utilizing Traffic Light Information for Reducing Idling at Red Lights and Improving Fuel Economy." Journal of Advanced Transportation **2018**.

26. Lu, G. (2007). Traffic light prediction system, Google Patents.

27. Lu, G. (2011). Traffic light prediction system, Google Patents.

28. Mahler, G. and A. Vahidi (2014). "An optimal velocity-planning scheme for vehicle energy efficiency through probabilistic prediction of traffic-signal timing." IEEE Transactions on Intelligent Transportation Systems **15**(6): 2516-2523.

29. Mahler, G., et al. (2015). Systems and methods for predicting traffic signal information, Google Patents.

30. Marshall, A. (2016) Enlighten App Uses AI to Predict When Lights Will Turn Green. Enlighten App Uses AI to Predict When Lights Will Turn Green

31. Mayank, K. (2015). "Idealing of Vehicle at Traffic Signals Lead to Fuel Wastage and Emission." International Journal of Science Technology and Engineering **1**(11): 63-67.

32. Müller, J. and K. Dietmayer (2018). Detecting traffic lights by single shot detection. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE.

33. Nikhila Haridas, S. S. (2018, April 27, 2018). "Traffic Light Detection Using the TensorFlow Object Detection API." Retrieved February 20, 2019, 2019, from https://software.intel.com/en-us/articles/traffic-light-detection-using-the-tensorflow-object-detection-api.

34. Parida, P. and S. Gangopadhyay (2008). Estimation of fuel loss during idling of vehicles at signalized intersections in Delhi. Journal of Indian Roads Congress.

35. Rouse, M. (2017) vehicle to infrastructure (V2I or v2i). vehicle to infrastructure (V2I or v2i)

36. Sarkis, A. (2017). "Self-Driving Cars: Implementing Real-Time Traffic Light Detection and Classification in 2017." Self-Driving Cars: Implementing Real-Time Traffic Light Detection and Classification in 2017 https://medium.com/@anthony_sarkis/self-driving-cars-implementing-real-time-traffic-light-detection-and-classification-in-2017-7d9ae8df1c58 Accessed January 26, 2019 2019.

37. Srivastava, V. (2018). "Self Driving Vehicles: Traffic Light Detection and Classification with TensorFlow Object Detection API." Self Driving Vehicles: Traffic Light Detection and Classification with TensorFlow Object Detection API https://medium.com/@UdacityINDIA/self-driving-vehicles-traffic-light-detection-and-classification-with-tensorflow-object-detection-d6a4d25e99c2 Accessed January 15, 2019 2019.

38. Staff, G. T. N. (2014) Traffic App Predicts Green Lights. Traffic App Predicts Green Lights

39. Tielert, T., et al. (2010). The impact of traffic-light-to-vehicle communication on fuel consumption and emissions. 2010 Internet of Things (IOT), IEEE.

40. Verma, M. P. and M. V. B. P. Singh "Traffic Light Recognition System: A Computer Vision based Approach."

41. Vishal, K., et al. (2019). Traffic light recognition for autonomous vehicles by admixing the traditional ML and DL. Eleventh International Conference on Machine Vision (ICMV 2018), International Society for Optics and Photonics.

42. Williams, B. (2017) Cadillac tech 'talks' to traffic lights so you don't run them. Cadillac tech 'talks' to traffic lights so you don't run them

43. Schlanger, Z. (2015, February 13, 2015). "Drivers Exposed to 29 Times More Air Pollution While Stopped At Red Lights, Study Finds." Tech & Science. 2020, from https://www.newsweek.com/drivers-exposed-29-times-more-air-pollution-while-stopped-red-lights-study-306698.

44. Parrado, N. and Y. Donoso. (2015, March 31, 2015). "Congestion Based Mechanism for Route Discovery in a V2I-V2V System Applying Smart Devices and IoT. " Retrieved March 3, 2019, 2019, from https://www.mdpi.com/1424-8220/15/4/7768/htm.

45. Griffin, D. (2012, August 21, 2012). "MIT project uses smartphones to collect traffic light data." Retrieved March 3, 2019, 2019, from https://www.pocketgpsworld.com/MIT-project-uses-smartphones-to-collect-traffic-light-data-1055.php.

46. (2017, April 19, 2017). "Traffic Light Countdown Timer." Retrieved March 3, 2019, 2019, from https://www.sphere.hr/en/products/traffic-light-countdown-timer/.